НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *004.94; 004.042*

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Управління процесами збереження даних у сховищах центру обробки даних*

Виконав: студент *VI* курсу групи ІС-61м
(шифр групи)

Сягайло Тетяна Андріївна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *доц., к.т.н., доц. Жаріков Е.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

РЕФЕРАТ

Магістерська дисертація: 99 с., 35 рис., 12 табл., 1 додаток, 52 джерела.

Актуальність. На сьогодні все більше даних зберігаються на електронних пристроях, таких як HDD, SSD або оптичні накопичувачі. Але зі збільшенням інформації постає проблема управління процесами збереження даних, так як великі підприємства можуть використовувати сотні пристроїв для збереження, до яких одночасно необхідно отримувати доступ деякій кількості користувачів. Одним із способів уникнення втрати інформації, працівниками в сфері IT було запропоновано можливість налаштування сховищ.

Сховища призначені для організації процесів довгострокового збереження даних в системах автоматизованої обробки інформації. Даний вид збереження інформації дозволяють користувачам зменшити час на резервне копіювання даних, надають можливість користуватися файлами одночасно деякою кількістю користувачів. Перевагами їх використання є покращення швидкості доступу до необхідних файлів; можливість збільшення обсягу пам’яті, яку можна використовувати для збереження даних шляхом додавання додаткових пристроїв до сховищ; підвищення продуктивності роботи з даними; масштабованість та можливість створення резервних копій файлів. Сховища можуть використовувати як на великих підприємствах, так і в домашніх.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Розробка та впровадження системи управління IT-інфраструктурою з консолідованими інформаційно-обчислювальними ресурсами» (№ 0115U000322).

Метою дослідження є підвищення продуктивності під час роботи з даними в гіперконвергентних хмарних системах.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- проаналізувати існуючі рішення систем збереження даних;

- створити математичні моделі для управління міграцією і реплікацією даних;
- оформити вхідні даних для роботи системи;
- розробити алгоритми для управління процесами збереження даних;
- розробити середовище для впровадження розроблених алгоритми;
- реалізувати розроблені алгоритми в системі моделювання;
- проаналізувати результати роботи алгоритмів.

Об’єктом дослідження є процес управління збереженням даних при роботі зі сховищами даних.

Предметом дослідження є моделі для розподілення ресурсів на сховищах та методи управління процесами збереження даних.

Методами дослідження є методи і алгоритми управління процесами збереження даних.

Наукова новизна отриманих результатів. В представленій роботі досліджено спосіб управління процесами збереження даних на сховищах використовуючи методи реплікації та міграції даних в гіперконвергентних хмарних системах.

Публікації. Матеріали роботи опубліковані у тезах 10-ї Всеукраїнської науково практичної конференції «Комп’ютерні інтелектуальні системи та мережі» 2017 [1]; в публікації Міжнародної наукової конференції «Актуальні наукові дослідження в сучасному світі» [2, 4]; в тезах наукової конференції «Інформатика та обчислювальна техніка – ІОТ-2018» [3].

ФІЗИЧНІ СЕРВЕРА, БАГАТОРІВНЕВІ СХОВИЩА, ГІПЕРКОНВЕРГЕНТНІ СИСТЕМИ, БЛОКИ ДАНИХ, РЕПЛІКАЦІЯ ДАНИХ, МІГРАЦІЯ ДАНИХ

ABSTRACT

Master dissertation: 99 p. , 35 fig., 12 tab., 1 app., 52 sources.

Topicality. Today, more and more data is stored on electronic devices such as HDDs, SSDs or optical drives. But with increasing information, there is a problem managing data storage processes, as large enterprises can use hundreds of storage devices, which simultaneously need to access some users. One way to avoid losing information was to offer IT staffers the ability to set up repositories.

The repositories are designed to facilitate management while working with data. This type of save information allows users to reduce the time to back up data, provide the ability to use files at a time with a number of users. Advantages of using them are to improve the speed of access to the necessary files; the ability to increase the amount of memory that can be used to store data by adding additional devices to the repositories; increase data productivity; scalability and the ability to back up files. Storages can be used both at large enterprises and at home.

Relationship of work with scientific programs, plans, themes. The research was carried out at the Department of Computer-Aided Management And Data Processing Systems of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» within the theme «Development and implementation of an IT infrastructure management system with consolidated information and computing resources» (№ 0115U000322).

The aim of the research is increasing performance of data storage within hyperconvergence systems.

To achieve the goal, the following tasks must be performed:

- analyze existing solutions for data storage systems;
- develop mathematical models for allocating data management resources;
- configure the input data for data keeping;
- develop algorithms for managing data storage processes;
- develop an environment for implementation of the algorithms;

- implement developed algorithms;
- analyze the results of work.

The object of research is process of managing data keeping with data storages.

The subject of research is a model for distributing resources in storage and methods for managing data storage processes.

Research methods is method of managing data keeping with data storages.

Scientific novelty of the obtained results. In the presented work the method of managing data storage processes in the repositories is investigated using data replication and migration methods in hyperconverting cloud systems.

Publications. The materials of research are published in theses of the 10th All-Ukrainian Scientific and Practical Conference «Computer Intelligent Systems and Networks» 2017 [1]; presented at the International scientific conference «Actual research in the modern world» [2, 4]; published in theses of scientific conference «Informatics and Computer Science – IOT-2018» [3].

PHYSICAL SERVERS, MULTITIERING STORAGES, HYPERCONVERGENCE SYSTEMS, DATA BLOCKS, DATA REPLICATION, DATA MIGRATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ФС	–	фізичний сервер.
ВМ	–	віртуальна машині.
ЦОД	–	центр обробки даних.
HDD	–	англ. Hard Disk Drive – жорсткий магнітний диск.
SSD	–	англ. State Solid Drive – твердотілий накопичувач.
SAN	–	англ. Storage Area Network - система сховищ зберігання даних.
NAS	–	англ. Network-Attached Storage - мережі зберігання даних.
DAS	–	англ. Direct-Attached Storage - сховища з прямим підключенням
БС	–	багаторівневе сховище.
Міграція	–	переміщення даних між різними рівнями сховища.
Реплікація	–	створення копій даних одного фізичного серверу на іншому для уникнення можливості їх втрати.
ЦП	–	центральний процесор

ВСТУП

З розвиненням IT-індустрії, все більше та більше інформації зберігають на електронних пристроях. Але зі збільшенням кількості інформації постає проблема її збереження, так як до одних тих самих даних одночасно можуть отримувати доступ декілька користувачів, через це може збільшуватись час очікування виконання операцій над даними. На сьогодні збереження даних відбувається використовуючи декілька видів накопичувачів, такі як HDD, SSD та оптичні накопичувачі (DVD, CD). Останні два види для збереження великого об’єму даних є не дуже зручно, через це їх не використовують на сховищах даних через невеликий об'єм даних, які можуть на них зберігатись. Через те, що об’єм інформації з кожним роком зростає, для підприємств стає необхідним налаштовувати сховища, які будуть складатися з декількох сотен накопичувачів, які необхідно використовувати в залежності від характеристик та продуктивності. При виборі способу налаштування роботи сховища на підприємствах, на сьогодні, класифікують власні локальні та мережеві сховища, збереження даних на яких відбувається за деяким заданим набором правил.

До мережевих сховищ відносять використання хмарних сховищ або вже існуючих технологій збереження даних. Якщо розглядати існуючі на сьогодні хмарні сховища, то варто виділити наступні:

IDrive. Якщо розглядати дану систему для збереження даних, то IDrive надає можливість наступні особливості користування сервісом з декількох пристроїв одночасно; швидкого резервування та вилучення даних при необхідності; управління резервуванням даних та поновлення їх при втратах; стиснення файлів, для оновлення резервних копій; швидкий та зручний перегляд інформації змін на сховищах; збереження видалених файлів протягом 30 днів з можливістю поновлювати; та інші [5].

Даний вид сховищ надає високу пропускну спроможність для отримання даних, приблизно 4.5 Mbps та для їх збереження - 7Mbps [6].

Microsoft OneDrive. Програмний продукт компанії Microsoft розроблений у 2007 році. Хмарне сховище призначене у використані як у великому бізнесі, так і на

невеликих підприємствах. При роботі зі сховищем до переваг варто віднести наявність двохкрокової авторизації, синхронізація на рівні блоків та можливість отримання доступу до даних одночасно з декількох електронних пристроїв. При користуванні наявна можливість отримання доступу до історії роботи з файлами [7].

Результати перевірки швидкості завантаження файлів до сховища показали, що при швидкості 10 МБ/с завантаження 1ГБ даних відбувається протягом 30 секунд, що є непоганим показником.

DropBox. Випуск продукта відбувся у 2007 році компанією Dropbox Incorporation. На березень 2016 року кількість користувачів сервісом була 500 млн, яка зросла на 20% порівняно з червнем 2015 року [8]. Хмарне сховище надає можливість відновлення даних після видалення протягом 30 днів, отримання доступу до системи з декількох пристроїв одночасно та можливість отримання історії роботи з файлами та змін у системі. З випуском нової версії продукту продуктивність зростає через збільшення пропускної спроможності. Порівнюючи з безкоштовним об'ємом пам'яті, який надається користувачам, Dropbox надає тільки 2ГБ, в той час як IDrive та OneDrive надають до 5 ГБ.

Google Drive. Програмний продукт розроблений компанією Google та представлений світу 2012 році. На березень 2017 року кількість користувачів сервісом була приблизно 800 мільйонів чоловік [9]. Перевагами представленого сховища є можливо найбільш широкий вибір серед додатків, які можна додати при користуванні сховищем, також користувачі сервісу можуть надавати доступ до файлів як деяким користувачам, так і тим, хто мають посилання на даний файл. При надані доступу до файлів є вибір між можливостями користувача, який також має доступ до цього файлу, він може тільки переглядати документи, коментувати їх або редагувати. Недоліком є те, що користувачі системи, які не авторизовані в системі, не можуть редагувати файли, а тільки їх переглядати.

В статті [10] авторами перевірено час запису та читання файлу з чотирьох хмарних сховищ, які в результаті показали, що найменший час завантаження даних на сховища було витрачено в Google Drive, а найменший час завантаження даних зі

сховищ є IDrive. В середньому всі описані вище сховища показали приблизно однаковий результат.

Apple ICloud Drive. Програмний продукт розроблений компанією Apple та випущений у 2011 році. Доступ до сховищ даних мають лише користувачі операційних систем MacOS, iOS та Windows. В 2016 році було опубліковано, що загальна кількість користувачів цього сховища складає 782 млн [11]. Для безкоштовного користування сервісом користувачам надається можливість збереження даних на 5 ГБ.

Серед переваг програмного продукту ICloud є можливість користування однією інформацією одночасно з декількох електронних пристроїв, пошук файлів в залежності від пріоритету. Збереження даних на ICloud відбувається за допомогою використання ключ-значення (key-value), та значення ключа не повино перевищувати розмір в 64 Кб [12].

SugarSync. Віртуальна хмара розроблена компанією SugarSync у 2009 році. Перевагами при використані даного сховища є отримання доступу до файлів з декількох електронних пристроїв, збереження файлів великих розмірів, підтримка великої кількості операційних систем, наявність шифрування даних при збережені та відправки, наявність отримання інформації про останні зміни файлів, але лише на останні 5 змін. Але до недоліків користування цією віртуальною хмарою є не дуже швидка пропускна спроможність при роботі з нею. При завантажені файлів з приватної хмари розміром 1 Гб пропускна спроможність в середньому 17 Мб/сек, в той час як на завантаження файлів на приватну хмару близько 93 Мб/сек. Сховища зберігання даних розташовані на території Сполучених Штатів Америки [13, 14].

BackBlaze. Хмарний сервіс, розроблений компанією BackBlaze в 2007 році, поділяється на B2 Cloud Storage - об’єктне сховище даних, та Computer BackUp - утиліта, яка допомагає створювати користувачам резервні копії. Перевагами BackBlaze є створення резервних копій даних, до недоліків варто віднести неможливість поновлення даних після видалення протягом деякого проміжку часу. Резервні копії створюються на документи, фотокартки, будь-які формати даних, але не створюються на тимчасові файли та деякі системні, які не є важливими для

постійного зберігання. З метою підтримки можливості резервного копіювання файлів розміром до 90 Гб, сервіс передбачає створення резервної копії даних через 4 дні [15].

Carbonite. Програмний продукт розроблено у 2005 році компанією Corbonite Inc. Сервіс налаштований для надання можливості створення резервних копій в режимі онлайн. До переваг представленого продукту варто віднести необмежений об'єм даних для резервування, отримання віддаленого доступу до файлів. До недоліків варто віднести обмежену кількість даних для видалення та повільна швидкість завантаження файлів на сервіс для зберігання. При завантажені файлів розміром 1 Гб час виконання операцій може досягати 1 години, а приблизна швидкість передачі даних є 10 Мб/сек. Хоча слід зауважити, що при роботі з даними поточного сервісу час в середньому не перевищує середній час при роботі з іншими хмарними сховищами [16].

Використання хмарних сховищ для деяких підприємств може виявитись надто затратним. Також, державне регулювання вимагає від підприємств налаштовувати власні сховища. Таким чином, одним із варіантів організації сховища є використання систем NAS, SAN та DAS.

SAN сховища являють собою взаємодію хостів, елементів та накопичувачів, необхідних для збереження даних. Даний тип сховищ базується на використані протоколів Fibre Channel. В мережах SAN збереження даних відбувається блоками. До переваг використання даних сховищ варто віднести можливість полегшеного отримання доступу до ресурсів з будь-якого серверу, масштабованість даних мереж, більша продуктивність, наявність синхронизації для відмовостійкості даних, покращення ефективності зберігання, захисту та безпеки даних. В мережах SAN легкий процес адміністрування, що полегшує процес керування великою кількістю пристроїв для зберігання інформації. Найчастіше даний вид мереж використовується середнім і крупним бізнесом [17].

Іншим способом зберігання даних є використання сховищ типу NAS, які надають доступ до файлів в певній мережі, використовуючи TCP/IP підключення. Класифікація даних сховищ залежить від розміру підприємства. Для великих

підприємств – це можливість надання доступу до збереження великої кількості даних та кластеризації. Для середнього бізнесу – ємність сховищ складається тільки з декількох сотен терабайт, дані сховища не мають можливості кластеризації, але в них надається можливість створювати резервні копії файлів. На відміну від SAN, в сховищах типу NAS використовується файлове збереження даних. Перевагами даних сховищ є легкодоступність ресурсів та можливість одночасного доступу до файлу декількох користувачів, масштабованість, можливість збільшення пам'яті та управління сховищами з серверів. До недоліків варто віднести вартість даних сховищ. В залежності від кількості пристроїв, які необхідні для зберігання даних, даний тип мереж може стати дуже витратними для використання [18].

Ще один спосіб зберігання даних представлений сховищами DAS, або сховища з прямим підключенням. При цьому до кожного серверу підключено свій пристрій для зберігання даних. Кожен DAS сервер має доступ тільки до свого сховища. Як і сховища SAN, DAS сховища зберігають дані блоками. Порівняно з NAS та SAN, DAS є менш дорогим рішенням. Недоліком використання даного типу сховищ є неможливість отримання доступу до даних з інших сховищ серверами та неможливість масштабування. Дані, які зберігаються, є доступні тільки в середовищі одного серверу [19]. Отже, використання даних мереж є необхідним у випадку, коли масштабованість досягається додаванням нових серверів, а доступ до файлів з інших серверів не передбачений архітектурою.

В статті [20] автором зроблено порівняння значень IOPS та пропускної спроможності даних DAS та SAN сховищ на запис та читання. В роботі використано сервер з підключенням до SAN та внутрішнім локальним дискам, обєднаним по схемі RAID1. Також, в роботі проведено тести з DAS диском. Результати тестування показали, що сховища SAN є більш продуктивними та різниця в результатах дуже сильно відрізняється.

Через постійне збільшення кількості даних та користувачів, головною метою використання сховищ є пошук таких способів розміщення даних на накопичувачах, які в результаті зменшуватимуть час пошуку необхідних даних та процес збереження даних на яких, дозволить уникнути втрату даних на сховищах.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ЩОДО НАЛАШТУВАННЯ РОБОТИ СХОВИЩ ДАНИХ

1.1 Огляд сучасних досліджень

На сьогодні існує багато способів налаштування сховищ для користування, але в незалежності від обраного способу головною метою роботи зі сховищами є швидке отримання доступу до даних.

В роботі [21] також представлено систему керування невеликими сховищами даних IOFlow. Система є програмно-визначеним сховищем (SDS або software-defined storage), в яких розміщення даних відбувається з виконанням деяких заданих бізнес-орієнтованих правил. Створена система використовує політику e2e(end-to-end) при роботі з файлами та представила можливість контролю та планування роботи з даними, що повино полегшувати можливість роботи з системою.

Цілями створеної системи постала:

− покращення продуктивності під час наявності часу затримки;

− швидкорегульованість, масштабованість під час управління роботи зі системою;

− відсутність внесення змін у конфігураційних файлах додатків, які використовують дану систему або на віртуальних машинах під час налаштування.

Результати роботи представлені у вигляді графіків на яких можна побачити значення пропускної спроможності та затримки в залежності від кількості запитів, які направлено до сховища. Також показано, що при використані створеної системи значення продуктивності покращилось. Це видно з графіків залежності часу затримки від кількості запитів, використовуючи дану систему та без неї. Також, представлено графічно залежність пропускної спроможності від розміру даних з якими необхідно працювати для записуючих пристроїв SSD та HDD з використанням запропонованої системи IOFlow та без неї, з яких можно побачити збільшення значення пропускної спроможності зі збільшенням об’єму даних над якими необхідно виконати операції запису/читання.

В роботі [22] представлено систему для надання доступу до даних декільком користувачам одночасно, яка отримала назву Retro. Розроблена система складається з множини правил, які необхідні для покращення продуктивності в розподілених системах.

Архітектура Retro складається з процесів, ресурсів, місць в яких відбувається процеси виконання операцій над ресурсами та контролера, який містить містить правила необхідні для роботи системи. Результати використання представленої системи показали непогану пропускну спроможність на дискових пристроях використовуючи представлену систему.

Авторами в роботі [23] розроблено систему керування сховищами даних IOStack, що є програмно-визначеним сховищем (SDS або software-defined storage) розміщення даних на об’єктних пристроях використовуючи деякі бізнес-орієнтовані правила. Сервіси запропонованої системи можно використовувати як незалежний компонент або як фреймворки.

Були представлені графіки для порівняння продуктивності під час отримання доступу до даних декількома користувачами при паралельного виконання декількох PUT-запитів в об’єктному сховищі OpenStack Object Storage або Swift та IOStack. Опираючись на представлені авторами дані, можно зробити висновок, що розроблена система надає збільшену пропускну спроможність пристроїв зберігання, ніж в сховищах Swift. Отримані результати показують, що значення пропускної спроможності використовуючи об’єктне сховище Swift є в три рази меншими, ніж при використані розробленого сховища.

Розроблена система є гнучкою та дозволяє легко та ефективно використовувати великий обсяг можливостей, який в результаті може покращити та полегшити процес роботи з об’єктами під час використання об’єктних сховищ.

В роботі [24, 25] сформулювано задачу вирішення проблеми оптимізації розміщення блоків в багаторівневих сховищах з урахування відомих технічних характеристик. Перевагами представленої авторами ідеї є не тільки покращення продуктивності роботи сховищ, а й збільшення швидкості отримання доступу до даних та зменшення кількості витрат електроенергії під час використання описаних

сховищ. Виконуючи деякі операції для роботи з даними доступ до яких необхідно отримати, автори запропонували спосіб для збереження електроенергії шляхом вимкнення сховищ, на яких не зберігаються дані. Збереження даних базується на концепції MAID(massive array of idle disks), тобто користування тільки тими дисками, які на даний момент використовуються. Дана концепція дозволяє мінімізувати витрати на електроспоживання. Авторами були представлені три моделі, які дозволяють покращити процес роботи зі сховищами даних, але більш цікава з них є кінцева, яка запропоновує мінімізацію енерговитрат, грунтуючись на показниках витрат енергії в кожному сховищі окремо.

Система зберігання даних розбита на декілька рівнів для збереження. Рівні збереження були обрані згідно моделі запропонованої SNIA(storage networking industry association) [26], яка пропонує розподіл зберігання даних на 4 рівнях, таких як рівень пристроїв зберігання, блочного рівня, файлового рівня та рівня додатку. При запиті доступу додатком, доступ до блоків відбувається через файловий рівень в якому доступ є віртуалізований, що означає відсутність інформації про існуючі сховища для зберігання даних. Для розміщення даних на різних сховищах представлено дві структури їх збереження: горизонтальне розбиття, яке полягає в розміщення даних в різних строках в різних таблицях баз даних та вертикальне розбиття - полягає в створені таблиць з меншої кількості стовбців та створення додаткових стовбців, для збереження даних, які залишились.

Результатом роботи авторами реалізовано генетичний алгоритм, для пошуку оптимального розподілу блоків на різних рівнях для зберігання даних.

В роботі [27] описану систему сховищ Violet яка забезпечує ефективне функціональне управління даними та представлена нова структура представлення файлів на дискових пристроях, яка отримала назву Fibonaccі Array. Програми були реалізовані на мові C++. Представлена система у випадку наявних змін у сховищах виконує ефективний запис на жосткий диск. Створена авторами система є розподіленою та складається з декількох рівнів, такі як: рівень користувача або рівень додатку; серверів, які мають доступ до жорстких дисків; та головний сервер, з якого відбувається управління кластеру.

Сервери можуть складатися з одного або декількох дисків та деякої енергонезалежної пам’яті. Для додавання нового серверу до системи необхідно передати інформацію про нього до головного серверу, таку як кількість пам’яті та кількість дисків. Після її перевірки головний сервер додасть його до кластеру. Створена система не залежить від виду інформації, яка буде зберігатись у ній.

В статті [28] авторами запропоновано схему для збереження даних на логічних гібридних пристроях. Результатом представленої схеми є покращення роботи програм та підвищення продуктивності під час роботи зі сховищами. Під час експерементів використано менеджер логічних дисків LVM(logical volume manager), який використовується при роботі в операційній системі Linux. Перевагами даної схеми є усунення обчислювальних затримок при пошуку таблиць відображення та можливість додавання та вилучення фізичних пристроїв для зберігання інформації за допомогою використання LVM.

Запропонована схема підтримує міграцію блоків між накопичувачами SSD(solid-state driver) та жорсткими дисками HDD та навпаки. У випадку видалення блоків при роботі з доданком та при бажані міграції видалених блоків стає неможливим отримання доступу до них на жорстких дисках та твердотілих накопичувачах.

Проведені авторами експерименти представили розподіл доступу до блоків при роботі з різними додатками на запис та читання інформації. Представлений графічний розподіл запитів на різну довжину послідовності блоків при використані додатків LibreOffice та Eclipse показав, що більш частіше відбувається запит на менші послідовності блоків, ніж на більш довші. Результати експериментів показали зменшення часу запуску додатків на 45% при мігрувані тільки даних, та на 56% у випадку міграції даних разом з метаданими. Порівнюючи дану схему зі DM-cache авторами виявлено прокращення показників при міграції блоків.

В роботі [29] запропонована організація пристроїв для зберігання даних SSD та HDD в гібридній системі. Авторами зроблено вимірювання продуктивності на основі, яких створено організацію даних в системі зберігання інформації.

Авторами спроектовано такий розподіл в гібридній системі, який виконує

доступ до даних паралельно з ціллю зменшення часу затримки при очікувані виконання операцій та перевірка цих правил шляхом створення порівняльних тестів та трасування при роботі з реально існуючими даними. В роботі використано програму Dbench[30] та NetApp[31] мета яких є моделювання робочого навантаження.

Результати отримані під час запуску тестів були зображені на графіках, з яких можна пробачити залежність продуктивності роботи пристроїв для зберігання даних від кількістю процесів, які намагаються отримати доступ до даних на пристроях. При використані правил запропонованих авторами в програмі Dbench, показав розподілення даних між SSD та HDD у відношені 70% приходиться на SSD та 30% - на HDD. В той час як розподілення робочого навантаження NetApp між SSD та HDD, в результаті показав, що використання правил призводить до розподілу навантаження у половиному відношені, тобто 50% приходиться на SSD та 50% - на HDD. На представлених результатах пропускна спроможність кожного з пристроїв збереження окремо показує значно нижчу пропускну спроможність при використані пристроїв одночасно та при збільшені кількості процесів, які виконуються. Затримка при використані обох пристроїв одночасно також є нижче, ніж при використані їх окремо.

В роботі [32] запропоновано спосіб розміщення різних типів даних на розподілених сховищах.

Під час виконання експерименту використовували 4 сервери, кожен з яких складається з 6 дисків; 4 клієнта; 2 перемикача: один з яких налаштований на роботу з клієнтами, інший з серверами. Клієнти та сервери підключені в системі за допомогою використання 10 Gb Ethernet.

Під час експерименту відбувалась передача даних на запис/читання файлів розміром по 10 Гб та перевірено зміни пропускної спроможності в залежності від кількості файлів з якими необхідно працювати. Так як кожен сервер має в наявності декілька пристроїв для запису, то представлені результати дають можливість зрозуміти, що в залежності від кількості файлів, які використовуються, як буде змінюватись пропускна спроможність для запису на один пристрій для збереження на

одному сервері та запису на декількох пристроях.

Отримані результати показали, що у випадку використання розміщення даних запропонованих авторами значення продуктивності зростає приблизно на 0.25, що є гарним показником для використання даної системи. Якщо брати до уваги відношення пропускної спроможності від кількості файлів з якою працюють, то у випадку виконання операцій по одному файлу на декілька серверів значення пропускної спроможності не відрізняється, хоча відбувається збільшення кількості файлів різниця між пропускною спроможністю є більш помітною.

Запропонована авторами модель, в результаті представляє покращену процедуру розміщення файлів в розподіленій системі, яка в результаті призводить до збільшення пропускної спроможності, що призводить до покращення продуктивності.

В роботі [33] авторами представлено модель, яка створювалась з метою покращення роботи з міграції даних в залежності від заданих параметрів, таких як характеристики додатку, види та типи даних з якими необхідно співпрацювати та терміни виконання завдань в сховищах даних з декількома рівнями збереження. Створена схема міграції даних отримала назву ADLAM.

В представленій роботі описано необхідність існування терміну міграції для даних в мережі при покращення продуктивності системи від існуючих можливостей зберігання та процес оптимізації міграції даних для створення адаптивного схеми міграції даних. Система складається з трьох рівнів збереження, таких як:

- низький рівень - складається з деякої кількості приладів для зберігання даних;
- середній рівень - відповідає за місця збереження даних, планування процесів міграції;
- високий рівень - представляє вид на віртуалізоване збереження даних.

Результатом створення представленої системи зменшення навантаженості пристроїв для зберігання та покращення масштабованості системи. Отримані результати тестування показують покращення продуктивності при використання запропонованої схеми міграції порівняно з вже існуючими моделями та схемами.

В роботі [34] описано процес налаштування розподіленої системи, робочий процес якої складається з рівню представлення(хости на Apache Http Server [35]), бізнес рівень (застосування розроблене на мові програмування Java, яка працює з запитами від користувачів та відповідає за запис та читання з баз даних) та рівень збереження даних(реалізовано з використанням баз даних MySQL).

Авторами представлено ілюстрацію роботи даної системи на 10 суміжних хмарних серверах та приклад конфігураційного файлу для впровадження даного робочого процесу. Конфігураційний файл створюється разом з налаштуванням системи та складається з усіх робочих процесів, які використовуються в системі. В залежності від створеного файлу визначається продуктивність та вартість створеної системи.

Представлений процес в результаті надає користувачам можливість розуміння налаштування розподілених сховищ, з покращеною продуктивністю, масштабованістю та рівням безпеки сховищ.

При збережені великої кількості даних стає вірогідність відмовостійкості системи у випадку перевантаження системи. В роботі [36] авторами представлено декілька моделей, розраховані формули для знаходження показників надійності системи та побудовані моделі відмовостійких систем в RAID масивах, а саме RAID-0, RAID-1, RAID-5, RAID-6, з можливістю створення резервних копій файлів та відображено блок-схему алгоритму для знаходження показників надійності дискових пристроїв.

Для знаходження параметрів системи, авторами запропоновано використання значень інтенсивностей відмов дисків, регенерації даних, помилок читання диску, помилок контролера та повного відновлення системи. Описані вище дані можно розрахувати за допомогою вже відомих параметрів пристроїв для збереження, таких як ємність дискових пристроїв, середня швидкість запису та читання. Представлена авторами формула для знаходження надійності системи є більш точною через використання значень інтенсивності помилок контролера та помилок зчитування диску.

Представлена модель впроваджена у роботу на ряді підприємств при обробці та

аналізі пристроїв для читання даних.

В роботі [37] представлено хмарний сервіс, який створювався з метою передачі даних для високопродуктивних обчислень(high perfomance computing). Сервіс налаштований на отримання даних від обчислень, які можуть бути результатами від моделювання та його задачею є надання доступу до цих даних користувачам. Передача даних відбувається частинами для покращення продуктивності.

Хмарний сервіс складається з трьох головних компонентів, один з яких відповідає за взаємодію клієнта з хмарними сервісами, інший - за місце зберігання та процес передачі даних та останній - за збереження інформації на сховищах.

Розроблена утіліта, на момент написання статті, реалізована для використання на платформі Microsoft Azure[38] та написана на мові програмування C#.

В роботі представлені результати порівняння вже існуючих способів передачі даних. Для тестування були створені 5 аккаунтів, які розміщувались в різних кінцях світу(3 - на території Сполучених Штатів Америки, 1 - на території Европи, 1 - в Азії). Та перевірено пропускну спроможність та час передачі даних запитами PUT та GET в дані регіони файлів розміром 4 МБ. Отримані результати показали, що вища пропускна спроможність є до пристроїв, які знаходяться на території Америки. Також, в роботі представлено результати передачі файлів використовуючи різні вже існуючі технології та результати яких показали, що при використані описаної системи швидкість на запис та читання даних є вищою порівняно з вже існуючими.

У роботі [39] авторами представлено чотири моделі для покращення процесу управління сховищами даних та описано методи управління сховищами. Вхідними параметрами даних моделей є:

- множина сховищ на яких зберігаються дані;
- множина користувачів, які намагаються отримати доступ до даних;
- множина файлів, які зберігаються на сховищах або повині бути збережені на них;
- множина рівнів пам’яті.

Описані в роботі математичні моделі поділяються на дві групи кластеризації ресурсів та управління ресурсами і націлені на мінімізацію витрат на користування

існуючими сховищами, покращення якості обслуговування та розподілення файлів між рівнями збереження даних.

Також, проведено тестування роботи в використані реалізованих генетичного та еврестичного алгоритмів. Результати показали, що при роботі з файлами розміром до 100 Кб алгоритми надали приблизно однакові результати, але на файлах більшого розміру показники роботи генетичного алгоритму є порівняно кращими з результатами еврестичного алгоритму, через те що останній робить перебір усіх варіантів розміщення з метою знаходження кращого за визначеними критеріями. В роботі були графічно були представлені усереднені результати збільшення значення часу на розподіл файлів між пристроями в залежності від кількості файлів, отримані під час виконання експериментів.

З отриманих результатів авторами зроблено висновки, що швидкість знаходження розв’язку еврестичним алгоритмом є кращою, але результат роботи генетичного алгоритму є більш близький до оптимального.

В роботі [40] запропоновано спосіб налаштування багаторівневих сховищ даних, які складаються тільки з SSD накопичувачами, через те, що даний тип накопичувачів є більш вигідним з точки зору швидкості роботи з даними пристроями на сховищах даних. Розподіл пристроїв між рівнями сховища відбувається в залежності від продуктивності накопичувачів та їх вартості, тобто пристрої в яких показники продуктивності та вартість є вищими знаходяться на вищих рівнях, а на нижчих дані зберігаються на більш дешевих SSD пристроях.

Представлена в роботі модель націлена на збільшення прибутків при збережені даних на сховищах. Дана мета досягається шляхом міграції даних між пристроями в залежності від вартості збереження даних на пристрої та вартості їх міграції на інші пристрої, збереження на яких матиме нижчу вартість, що в результаті призведе до зменшення витрат на використання представленого сховища.

Описаний в роботі алгоритм реалізує автоматизацію процеса міграції даних між рівнями, що призводить до зменшення часу затримки при виконані операцій, які відбуваються на сховищах. У випадку наявності необхідності для міграції даних, розраховується її вартість та знаходиться значення покращення продуктивності при

виконані представленої міграції. Якщо виконання міграції призводить до зменшення вартості використання сховищ, то її варто виконати.

Представлена система збереження даних реалізована з використанням VMware ESXi [41]. Авторами проведено ряд тестів, в залежності від завантаженості системи, перевірено значення пропускої спроможності та IOPS. Результати тестувань використання преставленого алгоритму показали високі значення усереднених показників пропускної спроможності, часу затримки та продуктивності на усіх рівнях сховища, також представлено вартість міграції при використані алгоритму та кількість міграцій даних.

Якщо розглядати ще один спосіб розгортання багаторівневих системи, то в роботі [42] запропоновано кластер, в якому на нодах відбувається збереження даних на двох рівнях. Рівні збереження даних поділені в залежності від показників продуктивності накопичувачів, на одному рівні розміщені SSD пристрої, на іншому - HDD. Збереження даних на SSD пристроях поділяється на два типи, перший з яких це збереження деяких даних локальних віртуальних машин, інший - зберігає дані, які розміщення на іншому ноді, даний спосіб збереження надає можливість запобігати можливість втрати даних, шляхом створення резервних копій на інших нодах. HDD рівень використовується для зберігання копій даних, які зберігаються на представленому ноді на SSD пристроях.

Також, в роботі представлено ряд процедур, які описують процес створення дублікатів даних та їх міграцію між нодами сховища для збереженя даних. Ще одна представлена процедура для роботи сховища направлена на відновлення даних на SSD накопичувачах у випадку втрати даних з них.

Реалізація даної системи відбувалась з використанням VMware ESXi [41]. Проведено порівняння при роботі системи без створення резервних копій даних, зі створенням та з використанням представленого способу розміщення даних на накопичувачах. Проведені авторами тестування розробленого алгоритму, показали непогані результати, у випадку втрати інформації на накопичувачах та проміжок часу її відновлення за допомогою даних, які зберігаються на інших нодах, пристроях та рівнях системи. Також, проведені дослідження визначення відношення наявності

різної кількості нодів на кластері та швидкість відновлення даних на них у випадку втрати.

1.2 Огляд аналогів на ринку схожих рішень або продуктів

На сьогодні ринок IT-інфраструктур розвивається і надає велику кількість продуктів для використання. При необхідності збереження даних на даний момент збільшується об’єм та кількість пристроїв, задача яких полягає в збережені даної інформації. При збільшені кількості пристроїв, необхідно зменшувати час очікування доступу до даних на них.

Якщо розглядати пристрої для накопичування даних, які на сьогодні пропонує ринок продуктів, то думка компаній поділилась на дві частини, одна з яких пропонує використання гібридних сховищ, які складаються з HDD та SDD накопичувачами та сховищ, які складаються тільки з флеш-пристроїв(all-flash storage). Виконаємо огляд існуючих на сьогодні технологій, які направлені на оптимізацію збереження даних на пристроях, для покращення процесу роботи сховищ даних, які використовують лише флеш-пристрої:

IBM's EasyTier[43]. Технологія запропонована компанією IBM з метою покращення продуктивності при роботі з багаторівневими системами сховищ. EasyTier пропонує використання декількох рівнів для збереження, таких як:

− Tier 0, який складається з флеш-пристроїв, які мають найкращі характеристики, такі як пропускна спроможність та частіше за все, вартість яких є найбільша.

− Tier 1 складається з флеш-пристроїв, які мають гірші характеристики, пристрої на рівні tier 0, але більше значення емності. Частіше за все, дані пристрої матимуть меншу вартість.

− Enterprise tier - рівень, який складається з Mdisk [44] пристроїв, які мають краще значення продуктивності.

− Nierline tier - рівень, який складається з Mdisk [44] пристроїв, які мають меншу вартість та більшу емність для збереження на них даних.

У випадку роботи з одним з типів MDisks пристроїв система перемикається в режим балансування, у випадку коли типи пристроїв є різними, то система міграції пристроїі між рівнями вмикається для оптимізації покращення процесу роботи збереження даних на сховищах.

Перевагами для використання даної системи для роботи з багаторівневими сховищами є:

− можливість вибору режиму при налаштувані системи, що призводить до збільшення продуктивності при роботі з флеш-накопичувачами, які мають кращу продуктивність порівнюючи з HDD пристроями.

− можливість отримання статистики роботи зі системою, в будь-який проміжок часу, використовуючи інструменти доступ до яких надає система.

До недоліків використання представленої системи є наявність обмежень на розміщення даних на пристроях та вимога використання на одному рівні пристроїв з однаковими характеристиками.

EMC's FAST[45]. Представлена технологія налаштування багаторівневих сховищ направлена на покращення продуктивності при роботі зі сховищами даних зі зменшенням витрат при використані них. Система налаштована на поділ сховищ на три рівні:

− Tier 0 - складається з флеш-пристроїв, які мають високу продуктивність.

− Tier 1 - рівень на якому знаходяться пристрої з непоганою продуктивністю, але нижчею ніж на першому рівні.

− Tier 2 - рівень, який складається з пристроїв з великою ємністю та які є найбільш дешеві.

Розподіл даних між рівнями сховища базується на результатах проведених досліджень, які показали, що тільки 5% даних продовжують використовувати після їх створення. В залежності від використання даних відбувається їх перерозподіл між рівнями:

− деякий відсоток даних, які використовуються часто зберігаються на верхньому рівні;

− дані, які використовуються не так часто на середньому;

− ті, які протягом довгого часу не використовуються - зберігаються на самих повільних пристроях останнього рівню.

Перевагами [46] використання даних сховищ є:

− наявність вибору накопичувачів (дана технологія дозволяє використовувати для налаштування сховищ не тільки SSD накопичувачів, а й HDD з різними характеристиками, в залежності від наявних пристроїв);

− можливість додавання додаткових технологій наявних у даного виробника при роботі зі сховищами даних для покращення продуктивності.

Pure[46]. Технологія налаштування сховищ, яка розроблена у 2009 році. При налаштувані та збережені даних на сховищах, дані можуть підвергатися дедублікації, яка необхідна для збільшення вільного об'єму для збереження даних на дисків шляхом стиснення даних, які вже на них зберігаються. Також, дані сховища пропонують можливість масштабованості сховищ, у випадку необхідності зміни архітектури сховищ.

Сховища, які дозволяє розгортати дана технологія знаходяться на одному рівні та повністю складаються з флеш-пристроїв, через великі значення показників продуктивності при роботі зі сховищами. Флеш-пристрої сховища використовують протоколи доступу NVMe.

HPE 3PAR[47]. Так як і Pure технології, система 3PAR не є багаторівневою і складається з використання лише SSD накопичувачів. До переваг використання даних технологій є можливість масштабованості системи, наявність прогнозування, швидкість доступу до даних, автоматичне створення резервних копій даних та безпека збереження даних, для уникнення ситуації їх втрати. Представлений спосіб налаштування сховищ націлений на використання на великих підприємствах. Системи надають можливість міграції даних між накопичувачами в залежності від стану в якому на даний момент вони знаходяться.

Якщо розглядати рішення на ринку продуктів, які пропонують використання гібридних сховищ, то варто звернути увагу на наступні рішення:

IBM Hybrid Storage[48]. Продукт розроблений компанією IBM. Як описано вище, тільки до п'яти відсотків даних, які зберігаються на сховищах є постійний доступ, усі інші знаходяться в стані невикористання. Опираючись на це компанія IBM розробила продукт в якому дані, які не використовуються зберігаються на більш повільних пристроях(HDD), а ті доступ до яких відбувається постійно зберігають на SSD накопичувачах. Перевагами використання гібридних сховищ IBM є можливість масштабованості, створення резервних копій файлів, висока продуктивність при роботі з даними та можливість управлінням процесом збереження та розміщення даних на пристроях.

Drobo[49]. Ще одним рішенням, який на сьогодні запропонований на ринку продаж є Drobo Hybrid Storages. Дане рішення надає:

- можливість масштабованості сховищ;
- можливість міграції даних між рівнями у випадку не використання протягом довгого часу;
- наявність захисту при збережені даних, у випадку втрати інформації на швидких рівнях сховища;
- наявність легкого процесу управління сховищами даних.

На сьогодні обробка і зберігання даних є важливою елементом під час введення бізнесу. Починаючи роботу зі сховищами, необхідно мати вже хоча б приблизне уявлення про інформацію та її розмір, яку необхідно буде зберігати на сховищах. Маючи великий об’єм даних, який необхідно опрацювати постає питання покращення продуктивності, яка призведе до підвищення швидкості доступу до даних та зменшення часу очікування виконання операції.

Доступ до даних може відбуватися постійно, такі дані варто зберігати на пристроях продуктивність яких є вище, частіше за все це флеш-пристрої SSD. Але для деяких з них доступ може бути дуже рідким, через це постає питання зберігання даного виду даних на пристроях продуктивність яких є нижчою. Якщо розглядати показники швидкості пристроїв під час роботи з файлами, то SSD пристрої працюють

набагато швидше. Було показано, що операції запису на SSD є набагато швидшими, ніж на HDD, та перевищують швидкість запису на останні в 80 разів, а при читанні в 50 разів [50]. Збереження тільки на пристроях SSD також є рішенням підвищення швидкості виконання операцій над даними, але постає проблема вартості їх придбання, бо порівнюючи пристрої однакової ємності, можна побачити, що вартість SSD в декілька разів перевищує HDD. Так середня вартість на 2017 рік 1 GB SSD - 17 центів, в той час як HDD — 6 центів за GB [51].

У випадку збереження даних на більш швидких пристроях до яких протягом довгого часу не відбувався доступ варто їх мігрувати на рівні збереження даних, які є більш дешеві для використання або навпаки, якщо дані знаходяться на більш повільних пристроях, але доступ до них протягом останього часу є постійним їх варто мігрувати на більш швидкі пристрої, так як це призводить до зменшення часу очікування користувачем.

Отже, для вирішення питання покращення продуктивності є знаходження оптимального розміщення файлів на пристроях збереження, яке зменшить час пошуку необхідних даних, зменшить значення енерговитрат та покращить пропускну здатність для роботи з файлами в середньому шляхом вибору сховищ з кращими характеристиками пристроїв.

Метою розробленого програмного продукту є підвищення продуктивності роботи сховища в процесах запису та читання даних ВМ в гіперконвергентних хмарних системах.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- проаналізувати існуючі рішення систем збереження даних;
- створити математичні моделі для розподілення ресурсів управління даними;
- оформити вхідні даних для роботи системи;
- розробити алгоритми для управління процесами збереження даних;
- розробити середовище для впровадження розроблених алгоритми;
- впровадити розроблені алгоритми;
- проаналізувати отримані результати роботи.

1.3 Висновок до розділу

В розділі проаналізовано сучасні дослідження в області налаштування систем для збереження даних, які показали, що проблемами при роботі зі сховищами є:

- втрата даних зі сховищ у випадку виходу з ладу ФС;
- час очікування виконання запитів на читання та запис даних зі сховищ.

Для вирішення поставлених проблем необхідно під час управління роботою сховищ налаштовувати можливість міграції даних на сховищі між рівнями та реплікації даних між ФС.

2 РОЗРОБКА МОДЕЛЕЙ ТА МЕТОДІВ РОЗМІЩЕННЯ ДАНИХ НА СХОВИЩАХ

З тим, що до сховищ повині входити велика кількість пристроїв, а основними проблемами при роботі з ними це:

- час доступу до даних віртуальними машинами;
- вірогідність втрати даних з пристроїв.

Ці проблеми можна вирішити за допомогою використання багаторівневих сховищ та реплікації даних між серверами сховища.

2.1 Багаторівневі сховища

2.1.1 Постановка задачі

При налаштувані декількох рівнів сховищ постає необхідність мігрувати між ними дані, для того щоб зменшити час очікування даних зі сховищ ВМ. Основними елементами, які необхідні для роботи сховищ, є дані, пристрої для збереження та сервери. До кожного сервера підключено декілька пристроїв, які розміщені на декількох рівнях, в залежності від їх продуктивності. Загальна кількість файлів, які зберігаються на поточному сервері, є *n*. Збереження файлів у сховищі відбувається блочно, розмір блоків залежить від умов використання і операційних систем. При постановці задачі розмір блока дорівнює 8 Кб. Кількість SSD на одному ФС – *m*, HDD – *c*. Для управління роботою сховища необхідно використовувати наступні характеристики:

F – множина файлів, які зберігаються на поточному ФС,

f_i – номер i-го файлу, $i = \overline{1,n}$,

d_i - розмір i-го файлу, $i = \overline{1,n}$,

$t = \frac{\sum_{i=1}^{n} d_i}{b}$ – кількість блоків на поточному ФС, де b – це розмір одного блоку,

b_j – номер блока, $j = \overline{1,t}$,

q_j – кількість разів доступу до j-го блоку протягом заданого часу (це може бути день, тиждень, місяць і т.д.), $j = \overline{1,t}$

B_{ij} – множина розбиття файлів на блоки (1 – коли j-ий блок є частиною i-го файлу, 0 – в іншому випадку), $i = \overline{1,n}, j = \overline{1,t}$,

S – множина SSD пристроїв,

s_k – розмір k-го SSD пристрою, $k = \overline{1,m}$,

x_{kj} – розміщення j-го блоку на k-ому SSD пристрої, $k = \overline{1,m}, j = \overline{1,t}$,

C_k - вартість використання k-го SSD пристрою, $k = \overline{1,m}$,

G_k - вартість зберігання блоку на k-му SSD пристрої, $k = \overline{1,m}$,

H – множина HDD пристроїв,

h_k – розмір k-го HDD пристрою, $k = \overline{1,c}$,

y_{kj} – розміщення j-го блоку на k-ому HDD пристрої, $k = \overline{1,c}, j = \overline{1,t}$,

D_k - вартість використання k-го HDD пристрою, $k = \overline{1,c}$,

g_k - вартість зберігання блоку на k-ому HDD пристрої, $k = \overline{1,c}$,

r_{vk} – вартість міграції блоку з v-ого пристрою HDD на k-ий SSD на, $k = \overline{1,m}, v = \overline{1,c}$.

2.1.2 Модель міграції даних між рівнями

Модель 1. Причиною для міграції між рівнями постає наявність/відсутність доступу до блоків даних протягом деякого часу. Через це необхідна максимізація середнього доступу до блоків, які розміщуються на рівні SSD пристроїв (2.1):

$$max \frac{\sum_{j=1}^{t} q_j x_{kj}}{\sum_{j=1}^{t} x_{kj}} \quad (2.1)$$

За умов виконання наступних обмежень:

− наявність вільного місця на накопичувачах, що може бути представлено наступними нерівностями (2.2 - для SSD, 2.3 - для HDD):

$$\sum_{j=1}^{t} x_{kj} b < s_k \quad (2.2)$$

$$\sum_{j=1}^{t} y_{kj} b < h_k \tag{2.3}$$

– на кожному рівні блок повинен зберігатись лише в одному екземплярі (2.4 - для SSD, 2.5 - для HDD):

$$\sum_{k=1}^{m} x_{kj} = 1 \tag{2.4}$$

$$\sum_{k=1}^{c} y_{kj} = 1 \tag{2.5}$$

– кожен з блоків належить тільки одному файлу (2.6):

$$\sum_{i=1}^{n} B_{ij} = 1 \, , j = \overline{1,t} \tag{2.6}$$

Модель 2. Під час збереження даних на пристроях необхідно знати вартість використання пристроїв. Знаходження вартості збереження даних на всіх SSD пристроях описується наступним виразом (2.7):

$$\sum_{k=1}^{m} C_k \tag{2.7}$$

Знаходження вартості збереження даних на HDD пристроях описується наступним виразом (2.8):

$$\sum_{k=1}^{c} D_k \tag{2.8}$$

Вартість зберігання блоків даних на *k*-ому SSD пристрої знаходиться за виразом (2.9):

$$\sum_{j=1}^{t} G_k x_{kj} \tag{2.9}$$

Вартість зберігання блоків даних на *k*-ому HDD пристрої знаходиться за виразом (2.10):

$$\sum_{j=1}^{t} g_k y_{kj} \tag{2.10}$$

У випадках, коли зберігання даних на пристроях з кращою продуктивністю може стати невигідним (дорогим), то необхідно мігрувати ці дані на інші рівні (наприклад HDD), де зберігання даних буде більш дешевим. Вартість міграції між двома пристроями різних рівнів розраховується наступним чином (2.11):

$$\sum_{v=1}^{c}\sum_{k=1}^{m} r_{vk} \tag{2.11}$$

Критерій мінімізації вартості збереження даних у сховищі з можливістю міграції даних між пристроями різних рівнів визначимо як:

$$\min(\sum_{k=1}^{m}(C_k + \sum_{j=1}^{t} G_k x_{jk}) + \sum_{k=1}^{c}(D_k + \sum_{j=1}^{t} g_k y_{jk}) + \sum_{v=1}^{c}\sum_{k=1}^{m} r_{vk}) \tag{2.12}$$

При умові виконання обмежень (2.2)-(2.6).

2.1.3 Метод міграції даних між рівнями

Так як не всі дані сховища використовуються постійно налаштовують БС (рисунок 2.1). Через що стає необхідною міграція даних, у випадку коли є в наявності велика кількість запитів до файлів протягом останнього часу. Для того, щоб на рівні швидких пристроїв(перший рівень) було в наявності місце необхідно підтримувати постійну частину вільного місця, яку можно використовувати для зберігання даних, які були збережені на рівень протягом останнього часу, яке потім звільняється за допомогою видалення файлів, які менше використовуються.

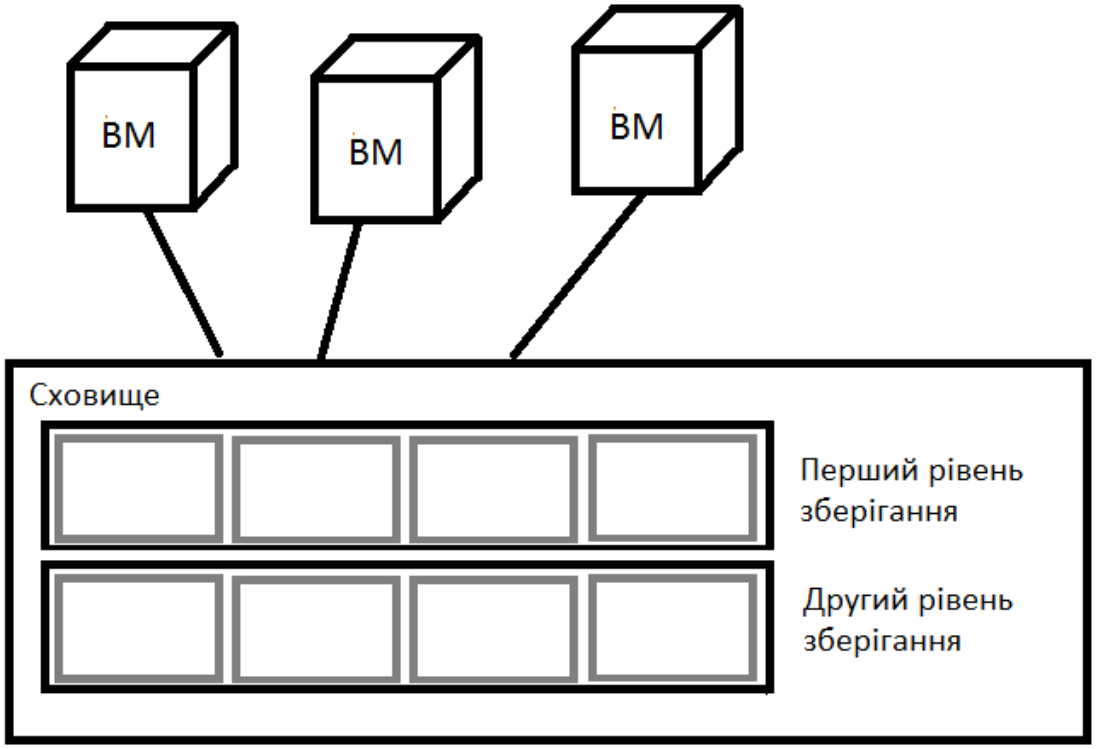


Рисунок 2.1 – Архітектура БС в межах одного ФС

Через це необхідно сортувати файли на сховищі в залежності від двох критеріїв:

- розмір файлу;
- кількість доступу до файлу.

Спосіб сортування файлів, які розміщуються на першому рівні сховища даних представлено в алгоритмі 2.1.

Алгоритм 2.1. Сортування файлів за критеріями розміру та кількості доступу до файлів

Вхідні дані: L^F - список файлів,

w_{size}– вага критерію розміру,

w_{access}– вага критерію кількості доступу до файлів

Вихідні дані: $newL^F$ -відсортований список за двома критеріями

1. Ініціалізація $newL^F \leftarrow$ NULL,
   $M^{F,Coef} \leftarrow$ NULL,
   $sizeL^F \leftarrow$ NULL,
   $accessL^F \leftarrow$ NULL;
2. $accessL^F \leftarrow$ відсортувати L^F за кількості доступу до файлів;
3. $sizeL^F \leftarrow$ відсортувати L^F за розміром;
4. **for** i = 0 **to** розмір L^F **do**

5. $M_i^{F,Coef} \leftarrow (L^F_i,\ w_{size} \times sizeL^F.index(L^F_i) + w_{access} \times accessL^F.index(L^F_i));$
6. **end for**
7. $M^{F,Coef}$ ← відстортувати $M^{F,Coef}$ за коефіцієнтом;
8. $newL^F$ ← список ключів з $M^{F,Coef}$;
9. **return** $newL^F$.

Нові файли зберігаються на першому рівні сховища та для початку додаються на вільне місце. Так як кількість вільного місця на першому рівні не повина бути менше визначеного порогового значення, файли, з найменшою кількістю доступів і найбільшого розміру необхідно видаляти з першого рівня поки на даному рівні кількість вільного місця не стане більше визначеного порогового значення, що представлено в алгоритмі 2.2.

Алгоритм 2.2. Видалення файлу з першого рівню для міграції

Вхідні дані: F – файл для міграції,

Вихідні дані: -

1. **while** $SIZE_F \geq s$ **then**
2. Видалити файл з L^F;
3. Видалити файл з першого рівня;
4. **end while;**
5. **return.**

Алгоритм міграції даних між рівнями показано в алгоритмі 2.3.

Алгоритм 2.3. Міграція даних з другого рівню на перший

Вхідні дані: F – файл для міграції,

s – кількість вільного місця на першому рівні

Вихідні дані: -

1. Ініціалізація L^F ← NULL;
2. L^F ← сортувати список файлів на першому рівні(алгоритм 2.1);
3. **while** $SIZE_F \geq s$ **then**
4. видалити файл з L^F;
5. **end while;**
6. Додати файл до першого рівня;
7. Звільнити місце на першому рівні (алгоритм 2.2)
8. **return**.

У випадку наявності доступу до файлів, міграція файлів, які знаходяться на другому рівні повина відбуватись, коли кількість доступу до файлу перевищує

деяке задане число. При перевищені даного числа запитів відбувається додавання файла на перший рівень.

2.2 Реплікація даних

2.2.1 Постановка задачі

Через обмеженість життєвого циклу пристроїв для збереження даних постає необхідність створювати копії даних (репліки) на пристроях зберігання даних інших серверів, в залежності від стану пристроїв поточного серверу і серверів реплікації на даний момент.

При виборі серверів, на які необхідно копіювати дані, необхідно звертати увагу на поточну завантаженість та кількість наявного вільного місця на HDD накопичувачах потенційних серверів-реплік, які необхідно формувати в критерії роботи серверів на поточний момент. Так як, на відміну від попередньої задачі, нам варто робити реплікацію між декількома серверами, то до використаних в попередньому пункті змінних варто додати коефіцієнт, який повинен описувати, який змінні якого саме серверу зараз використовуються. Нехай маємо *z* серверів. Для управління роботою сховища в режимі реплікації необхідно використовувати наступні характеристики:

C – множина серверів мережі,

O – множина файлів, які зберігаються на серверах (оригінал даних),

p – кількість первинних копій файлів, які зберігаються на серверах-джерелах;

H_l – множина HDD пристроїв на одному фізичному сервері, $l = \overline{1,z}$,

h_{lk} – розмір *k*-го HDD пристрою *j*-го серверу, $k = \overline{1,c}$, $l = \overline{1,z}$,

o_u – номер *u*-го файлу, $u = \overline{1,p}$,

e_u - розмір *u*-го файлу, $u = \overline{1,p}$,

w_{lk} – поточна завантаженість *k*-го HDD пристрою *l*-го серверу, яка залежить від поточної кількості реплікацій з інших серверів на поточний та запущених віртуальних машин, $l = \overline{1,z}$, $k = \overline{1,c}$,

R_{lu} – матриця розміщення файлів на серверах, включаючи реплікацію (1 – у випадку коли файли розміщені на сервері, 0 – коли копії файлу на сервері немає), $u = \overline{1,p}$, $l = \overline{1,z}$.

2.2.2 Модель реплікації даних між ФС

Для вибору місця створення реплікацій даних необхідно розрахувати два показники серверів:

- коефіцієнт вільного місця на серверах;
- коефіцієнт завантаженості поточного серверу реплікаціями даних.

Для знаходження коефіцієнту вільного місця на сервері (2.13), необхідно:

$$a_l = \frac{\sum_{k=1}^{m} h_{lk} - \sum_{u=1}^{p} d_u R_{ul}}{\max(\sum_{k=1}^{m} h_{lk} - \sum_{u=1}^{p} d_u R_{ul})} \quad (2.13)$$

Коефіцієнт завантаженості серверу реплікаціями (2.14):

$$W_l = \frac{\sum_{k=1}^{c} w_{lk}}{\max \sum_{k=1}^{c} w_{lk}} \quad (2.14)$$

Через це для пошуку місця на серверах для реплікацій даних необхідно знайти один або більше серверів-реплік за критерієм мінімуму даних коефіцієнтів (2.15):

$$\min(\frac{1}{a_l} + W_l), l = \overline{1,z} \quad (2.15)$$

При виконанні наступних обмежень:

- кількість даних, які зберігаються на сервері, включаючи реплікації даних з інших серверів, не повинні перевищувати загальний розмір накопичувачів даного сервера (2.16):

$$\sum_{u=1}^{p} d_u R_{ul} \leq \sum_{k=1}^{c} h_{lk} \quad (2.16)$$

- наявність копій файлів на серверах-репліках (2.17):

$$\sum_{l=1}^{z} R_{ul} > 1 \qquad (2.17)$$

– та при умові виконання обмежень (2.2)-(2.6).

2.2.3 Метод реплікації даних між серверами

Спосіб розподілення реплікацій даних між серверами представлено на рисунку 2.2.

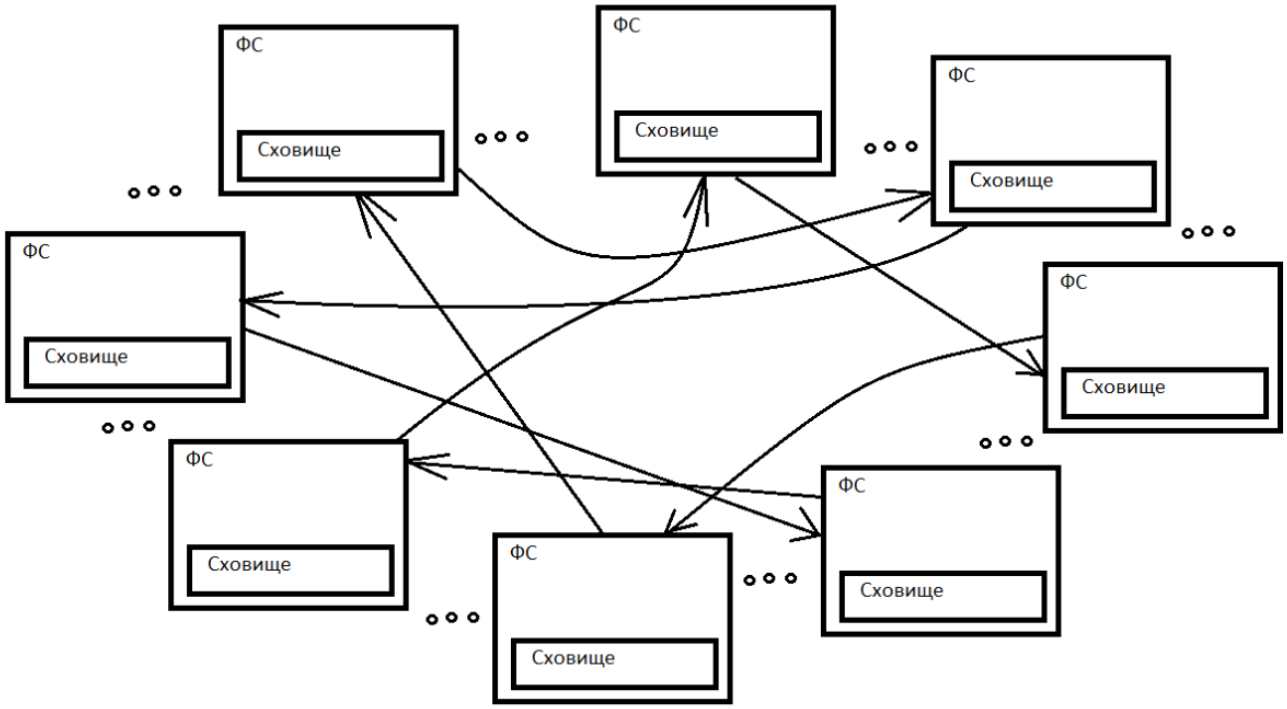


Рисунок 2.2 – Реплікація даних між декількома ФС

Вибір кращого серверу для реплікації даних повинен відбуватись в залежності від поточних показників серверів, які залежать від кількості вільної оперативної пам’яті, вільного місця на сховищі та пропускної спроможності серверу. Метод реплікації даних виконується в три етапи: визначення якості кожного серверу, обрання двох серверів з найкращими показниками за критеріями (2.13 -2.14), реплікація даних на обрані сервери. Для цього першим кроком є визначення кращих показників серверної системи, як показано в алгоритмі 2.4.

Алгоритм 2.4. Сортування серверів в залежності від кількості вільного місця на другому рівні

Вхідні дані: L^S- список серверів,

w_{size}– вага критерію вільного місця на другому рівні,

$w_{replica}$– вага критерію кількості серверів, дані з яких зберігаються на поточному сервері

Вихідні дані: $newL^S$ -відсортований список за двома критеріями

1. Ініціалізація $newL^S \leftarrow$ NULL,
   $M^{S,Coef} \leftarrow$ NULL,
   $sizeL^S \leftarrow$ NULL,
   $replicaL^S \leftarrow$ NULL;
2. $sizeL^S \leftarrow$ відсортувати L^S за кількістю вільного місця на серверах;
3. $replicaL^S \leftarrow$ відсортувати L^S за кількістю серверів, дані з яких зберігаються на поточному сервері;
4. **for** i = 0 **to** розмір L^S **do**
5. $M_i^{S,Coef} \leftarrow (L^S_i,\ w_{size} \times sizeL^S.index(L^S_i) + w_{replica} \times replicaL^S.index(L^S_i));$
6. **end for**
7. $M^{S,Coef} \leftarrow$ відстортувати $M^{S,Coef}$ за коефіцієнтом;
8. $newL^S \leftarrow$ список ключів з $M^{S,Coef};$
9. **return** $newL^S$.

Наступним кроком для необхідно обрати два сервери, які на поточний момент є кращим вибором для реплікацій даних з поточного серверу та виконати реплікацію даних з поточного серверу на два інші сервери показники, яких є вищі порівняно з іншими на поточний момент. Псевдокод виконання даної операції представлено в алгоритмі 2.5.

Алгоритм 2.5 – Реплікація дискових пристроїв на інші сервери

Вхідні дані: *data_disk* – дані з поточного серверу одного дискового пристрою

Вихідні дані: -

1. Ініціалізація *firstS* ← NULL,
   secondS ← NULL,
   $L^S \leftarrow$ NULL;
2. $L^S \leftarrow$відсортувати список серверів за двома критеріями (алгоритм 2.4);
3. $firstS \leftarrow L^S_1$;
4. Додати *data_disk* до *firstS;*
5. secondS← L^S_2;
6. Додати *data_disk* до *secondS;*
7. **return**.

2.3 Висновок до розділу

В розділі розглянуто три моделі управління процесами збереження даних на сховищах, дві з яких призначені для моделювання міграції даних між рівнями на одному фізичного серверу, та одна описує реплікацію даних між декількома фізичного серверу для уникнення можливості втрати інформації в хмарному сервісі.

Також, представлено алгоритми реалізації міграції та реплікації даних в сховищах а межах одного фізичного серверу та декількох відповідно. Алгоритм міграції даних, який було описано в розділі, переносить дані між рівнями сховища фізичного серверу, тільки у випадку наявності великої кількості доступа до файлу на операції читання/запису. В алгоритмі реплікації запропоновано обирати сервери-репліки даних в залежності від поточних їх показників, а саме кількості вільного місця на сховищах та кількість копій даних з інших серверів, які на даний момент вже зберігаються на сервері.

3 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Інформаційне забезпечення

3.1.1 Вхідні дані

Для формування вхідних даних для роботи зі сховищем використано застосування Process Monitor (рисунок 3.1) [52]. Process Monitor повертає перелік файлів, які використовуються при роботі з операційною системою Windows.

Time o...	Process Name	PID	Operation	Path	Result
8:16:12....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\dnsrslvr.dll	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\dnsrslvr.dll	SUCCESS
8:16:13....	svchost.exe	1712	ReadFile	C:\Windows\System32\dnsrslvr.dll	SUCCESS
8:16:15....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:24....	svchost.exe	1712	ReadFile	C:\Windows\System32\ncsi.dll	SUCCESS
8:16:24....	svchost.exe	1712	ReadFile	C:\Windows\System32\ncsi.dll	SUCCESS
8:16:24....	svchost.exe	1712	ReadFile	C:\Windows\System32\nlasvc.dll	SUCCESS
8:16:24....	svchost.exe	1712	ReadFile	C:\Windows\System32\nlasvc.dll	SUCCESS
8:16:26....	svchost.exe	1712	ReadFile	C:\Windows\System32\bcrypt.dll	SUCCESS
8:16:46....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:47....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:49....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:51....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:54....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS
8:16:56....	svchost.exe	1712	ReadFile	C:\Windows\System32\catroot2\{F750E6...	SUCCESS

Рисунок 3.1 – Процес роботи застосування Process Monitor

Аналізуючи показники використання ЦП та пам’яті, які показані на рисунках 3.2а-3.2б, можно зробити висновок, що найвищі показники є в період формування звітів. В даному прикладі ці показники знаходяться в допустимих межах.

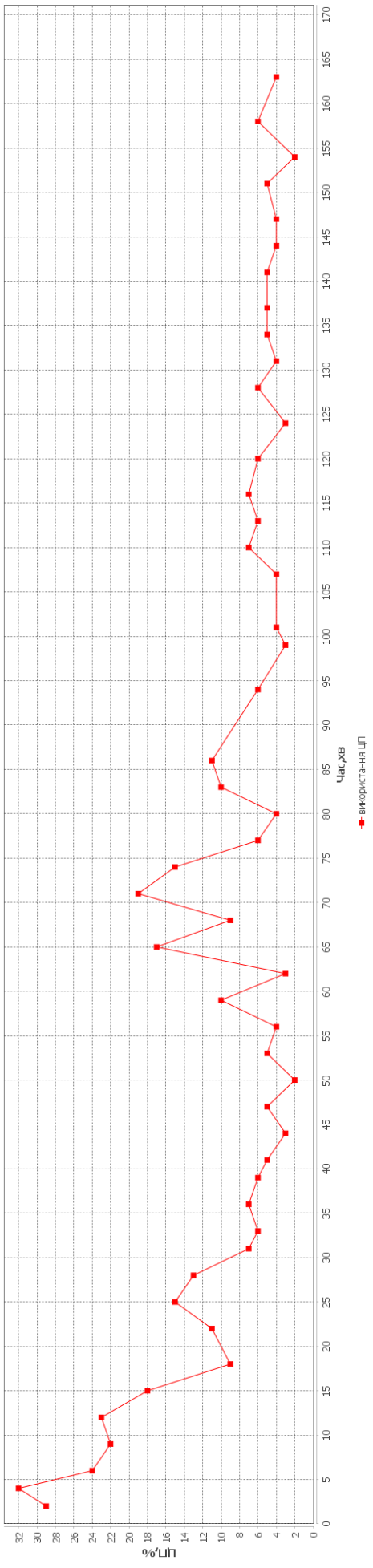


Рисунок 3.2а – Використання ЦП застосуванням Process Monitor

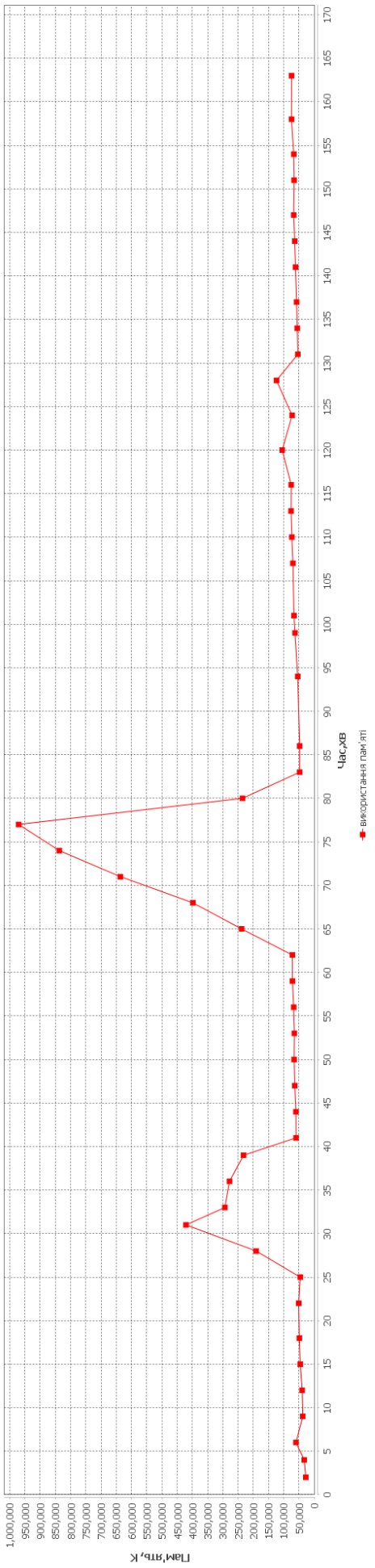


Рисунок 3.2б – Використання пам’яті застосуванням Process Monitor

Результатом роботи застосування є CSV-файл, в якому зберігається час, ідентифікатор процесу, шлях до файлу та інші дані, як показано на рисунку 3.3.

Time of Da	Process Na	PID	Operation	Path	Result	Detail	User
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, I	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, I	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:15.0	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, I	NT AUTHORITY\SYSTEM
01:15.0	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:15.1	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM

Рисунок 3.3 – Фрагмент звіту сформованого додатком Process Monitor при роботі з операційною системою Windows

Для визначення інтенсивності використання дискового пристрою сформований звіт було переформовано в залежності від кількості операцій читання і запису протягом заданого часу (рисунок 3.4).

Кількість операцій	Кількість секунд повторень даної кількості операцій
2	184
3	207
4	279
5	252
6	292
7	255
8	387
9	323
10	207
11	174
12	155
13	137
14	95
15	116
16	131
17	88
18	96
19	120
20	98

Рисунок 3.4. – Частота повторень кількості операцій з дисковим пристроєм

Розглядаючи дані, отримані в звіті, можно побачити, що кожної секунди кількість операцій читання і запису (IOPS), які виконуються з диском є різною. За час роботи експерименту кількість входжень зі швидкістю читання/запису в секунду більше 400 значно зменшується порівняно з кількістю операцій, кількість яких є нижчею. Таким чином, високе навантаження на диск в досліджуваній системі виникає рідко як представлено на рисунку 3.5.

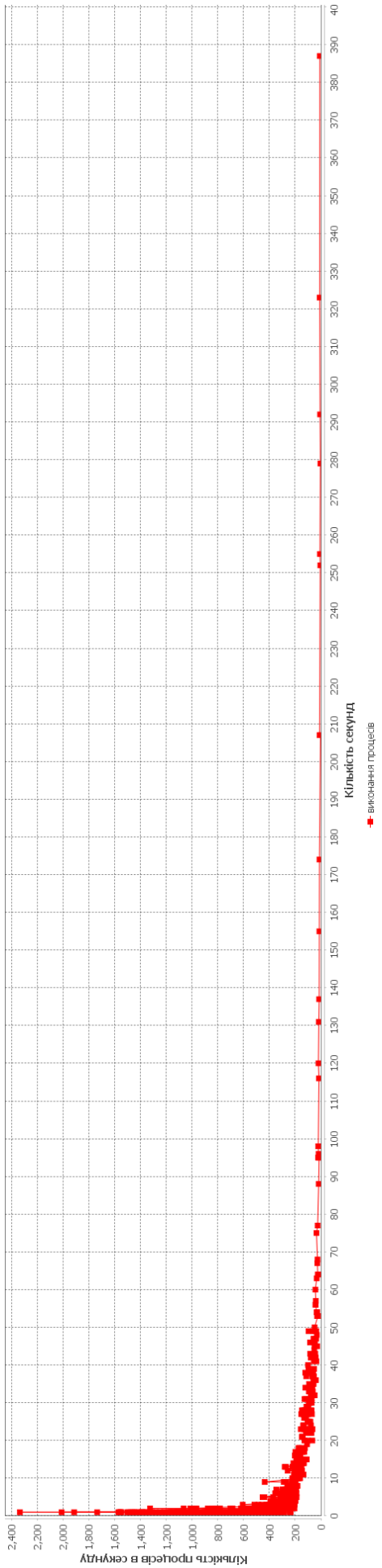


Рисунок 3.5 – Інтенсивність використання дискового пристрою кожної секунди за час експерименту

Після аналізу отриманих результатів сформовано вхідні дані для роботи з розробленим застосуванням, представлені на рисунку 3.6 та містять наступну інформацію про файл:

- шлях до нього;
- розмір;
- чи змінювався його розмір протягом часу роботи додатку;
- кількість запитів до файлу.

Path	Size	Change Siz	Count
"C:\Windo	0	FALSE	9
"C:\Windo	0	FALSE	22
"C:\Windo	14336	FALSE	2
"C:\Windo	249856	FALSE	394
"C:\Windo	0	FALSE	24
"C:\Windo	444752	FALSE	20
"C:\Windo	636048	FALSE	28
"C:\Windo	65536	FALSE	11
"C:\Windo	2319872	FALSE	91
"C:\Windo	491520	FALSE	24
"C:\Progra	16086	TRUE	1490
"C:\Progra	190	FALSE	13
"C:\Windo	115200	FALSE	2953
"C:\Windo	16896	FALSE	1681
"C:\Progra	19618	FALSE	438
"C:\Progra	25179	FALSE	104
"C:\Progra	32	FALSE	144

Рисунок 3.6 – Фрагмент вхідних даних

Більш детальне представлення формування вхідних даних для роботи програмного продукту представлено у графічних матеріалах.

3.1.2 Вихідні дані

Вихідними даними програмного продукту є час запису та читання даних при роботі з файлами на сховищах, показники поточної завантаженості дискових пристроїв та фізичних серверів при використані розроблених алгоритмів міграції та реплікації даних на сховищах.

3.2 Програмне та технічне забезпечення

3.2.1 Засоби розробки

Загальними засобами розробки даного програмного продукту є:

- Java - об’єктно-орієнтованій мові програмування, яка була випущена в 1995 році. При розробці продукту була використана 8 версія мови програмування.
- Apache Maven – фреймворк, який використовується для налаштування проектів за допомогою використання мови POM.
- Java Swing – бібліотека для створення графічного інтерфейсу.

3.2.2 Вимоги до технічного забезпечення

3.2.2.1 Загальні вимоги

Для правильної роботи програмного продукту необхідно мати наступне технічне забезпечення:

- ОС: Windows/Linux.
- Мова програмування: Java 8 або вище.
- Оперативна пам’ять: не менше 256 Мб.

3.2.3 Архітектура програмного забезпечення

3.2.3.1 Діаграма класів

Для формування вхідних даних розроблено програмне застосування структурна схема класів якого представлено на рисунку 3.7.

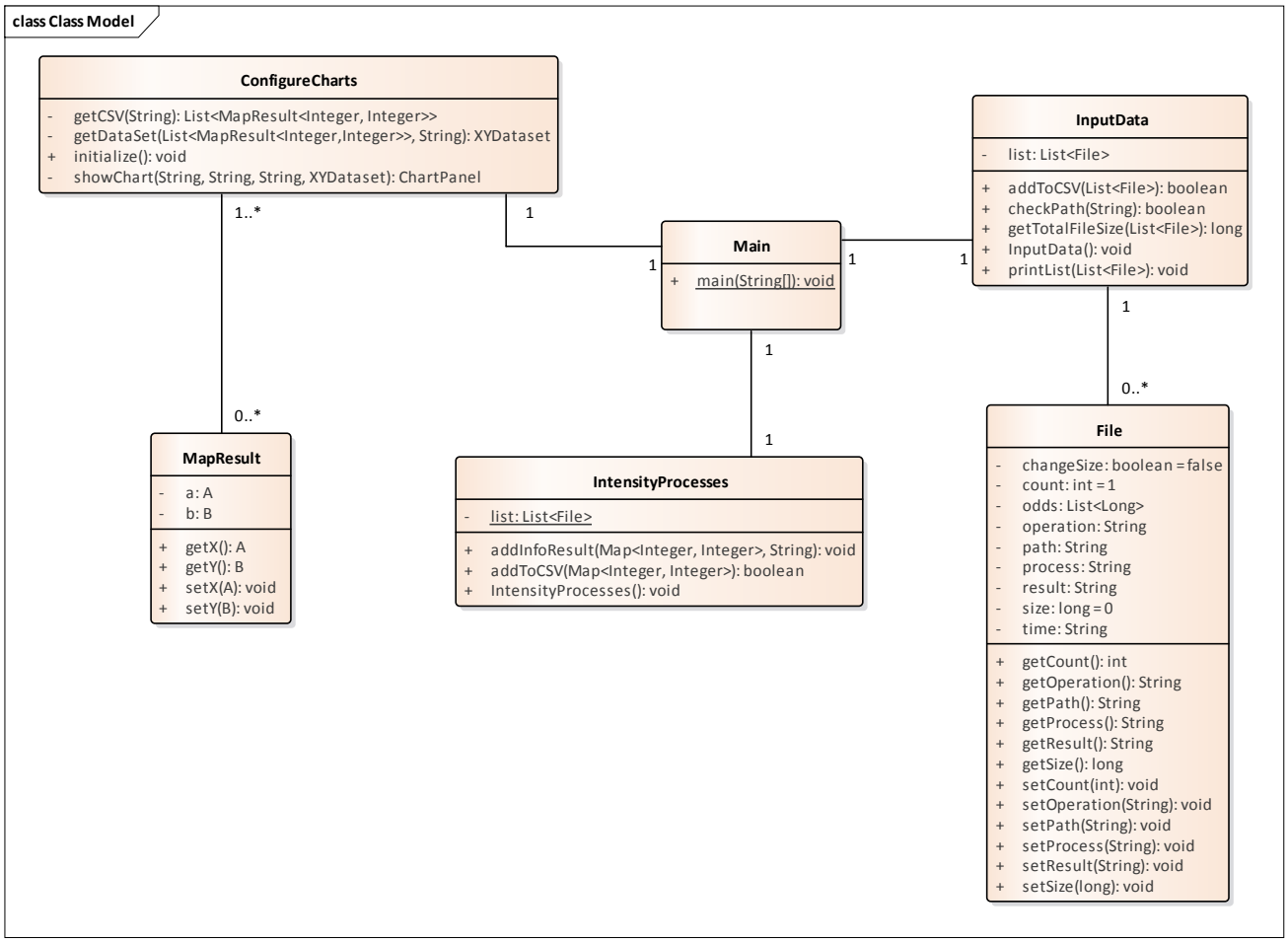


Рисунок 3.7 – Схема структурна класів для формування вхідних даних

Схема структурна містить 6 класів, таких як:

- Main – головний клас застосування;
- InputData – клас, перетворення даних зі сформованого звіту застосунку Process Monitor в CSV-файл вхідних даних;
- File – клас, збереження інформації про файл;
- IntensityProcesses – клас, який відповідає за формування звітів про використання даних дисковим пристроєм;
- ConfigureCharts – клас, який відповідає за формування графіків для аналізу використання ЦП та пам’яті програмним застосуванням Process Monitor;
- MapResult – клас, який зберігає дані, які необхідні для формування графіків дя аналізу даних.

Основна схема структурна класів розробленого програмного продукту представлена у частині графічного матеріалу.

Схема містить 1 інтерфейс:

- Disk– інтерфейс, який описує роботу дискового пристрою;

та 19 класів, а саме:

- Block – клас, який описує блок;
- Device – клас, який імплементує інтерфейс Disk;
- File – клас, який описує файл;
- Server – клас, який описує роботу серверу;
- VM – клас, який описує роботу ВМ;
- Storage – клас, який описує роботу БС;
- XYData – клас, який зберігає результати роботи серверів;
- Generate – клас отримання та формування вхідних даних системи;
- Migration – клас міграції даних між рівнями на сховищі;
- Replication – клас реплікації даних між ФС;
- Sort – клас сортування списку даних за двома критеріями;
- Simulation – клас управління процесом симуляції роботи системи сховищ даних;
- CustomOutputStream – клас для виводу інформації з консолі на графічний інтерфейс;
- Chart – клас оформлення даних та візуалізації їх на формах програмного продукту;
- MigrationPanel – клас відображення результатів міграції даних під час роботи програмного продукту;
- ReplicationPanel – клас відображення результатів реплікації даних під час роботи програмного продукту;
- ResultFrame – клас відображення результатів роботи застосування;
- MainFrame– клас реалізації графічного інтерфейсу для відображення процесу симуляції роботи сховища даних;
- Main – клас виводу результатів роботи в консолі для відображення процесу симуляції роботи сховища даних.

3.2.3.2 Специфікація функцій

Специфікація функцій для формування вхідних даних програмного продукту наведена в таблиці 3.1

Таблиця 3.1 – Специфікація функцій

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Main	Main	Головний метод програми	-	String[] args	Аргументи запуску програми
InputData	InputData	Конструктор класу InputData	-	-	-
InputData	addToCSV	Додати дані до CSV файлу	Boolean	List<File> list, String path	list – дані для збереження в CSV, path – шлях до файлу
InputData	checkPath	Перевірка наявності файлу в списку	Boolean	String path	path – шлях до файлу
InputData	printList	Вивід сформованого списку в консолі		List<File> list	list – дані для збереження в CSV
InputData	getTotalfileSize	Отримати загальний розмір файлів сформованого списку	long	List<File> list	list – дані для збереження в CSV
File	getId	Отримати унікальний номер кожного файлу	Integer	-	-
File	setId	Позначити унікальний номер файлу	-	Integer id	id-унікальний номер
File	getPath	Отримати шлях до файлу	String	-	-
File	setPath	Позначити шлях до файлу	-	String path	path – шлях до файлу
File	getOperation	Отримати операцію з файлом	String	-	-

Продовження таблиці 3.1

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
File	setOperation	Позначити операцію з файлом	-	String operation	operation – операція з файлом
File	getResult	Отримати результат виконання операції	String	-	-
File	setResult	Позначити результат виконання операції	-	String result	result – результат виконання операції
File	getSize	Отримати розмір файлу	String	-	-
File	setSize	Позначити розмір файлу	-	Long size	size – розмір файлу
File	getCount	Отримати кількість операцій над файлом протягом деякого часу	Integer	-	-
File	setCount	Позначити кількість операцій над файлом протягом деякого часу	-	Integer count	count – кількість операцій над файлом протягом деякого часу
File	getChangeSize	Отримати дані про зміну розміру файлу	boolean	-	-
File	setChangeSize	Позначити дані про зміну розміру файлу	-	Boolean changeSize	changeSize – чи змінювався розмір файлу
IntensityProcesses	IntensityProcesses	Конструктор класу IntensityProcesses	-	-	-

Продовження таблиці 3.1

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
IntensityProcesses	addInfoResult	Додати результати інтенсивності виконання операцій	-	Map<Integer,Integer> map, String path	map – результат інтенсивності виконання операцій кожної секунди, path – шлях до файлу
IntensityProcesses	addToCSV	Додати дані до CSV файлу	Boolean	List<File> list, String path	list – дані для збереження в CSV, path – шлях до файлу
ConfigureCharts	Initialize	Сформувати форму виводу графіків	-	-	-
ConfigureCharts	getDataSet	Сформувати дані для графіків	XYDataset	List<MapResult<Integer, Integer>> list, String title	list – дані для графіків, title – назва plot
ConfigureCharts	showChart	Сформувати панель з графічним інтерфейсом	ChartPanel	String title, String xLine, String yLine, XYDataset dataset	title – назва plot, xLine – назва осі X, yLine – назва осі Y, dataset – дані для графіків
ConfigureCharts	getCSV	Отримати дані з CSV	List<MapResult<Integer, Integer>>	-	-
MapResult	getX	Повернути значення осі X	X	-	-
MapResult	setX	Отримати значення осі X	-	X x	x – значення змінної на осі X
MapResult	getY	Повернути значення осі Y	Y	-	-
MapResult	setY	Отримати значення осі Y	-	Y y	y – значення змінної на осі Y

Специфікація функцій наведена в таблиці 3.2.

Таблиця 3.2 – Специфікація функцій

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
MainPanel	main	Головний метод програми	-	String[] args	Аргументи запуску програми
MainPanel	redirectSystemOutput	Перенаправлення виводу з консолі в JTextArea	-	JTextArea message	message - повідомлення
MainPanel	initialize	Ініціалізація форми	-	-	-
MainPanel	getCharts	Вивід графіків на графічний інтерфейс	-	-	-
Chart	getChartPanel	Шаблон формування графіків	ChartPanel	String title String xLine String yLine XYDataset dataset	title – назва графіку xLine – назва X осі yLine – назва Y осі dataset – дані для графіків
Chart	getDataSet	Сформувати дані для графіків	XYDataset	List<List<MapResult<Double, Double>>> map String title	map – дані для графіків title – назва прямої
Chart	saveChartAsImage	Збереження графіку в зображення	-	JFreeChart chart,String path	chart – графік для збереження, path- шлях до зображення
ResultFrame	ResultFrame	Конструктор класу ResultFrame	-	-	-
ResultFrame	start	Відкриття форми виводу результатів дослідження	-	List<Server> list	list – список серверів системи

Продовження таблиці 3.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
ResultFrame	initialize	Ініціалізація форми виводу результатів дослідження	-	-	-
MigrationPanel	MigrationPanel	Конструктор класу MigrationPanel	-	Integer serverId, XYData write, XYData read, XYData migration, Double freeFasterTier	serverId – номер серверу, write – результати запису даних, read – результати читання даних, migration – результати міграції даних, freeFasterTier– кількість вільного місця на першому рівні
MigrationPanel	initialize	Ініціалізація панелі виводу результатів міграції	-	-	-
ReplicationPanel	ReplicationPanel	Конструктор класу ReplicationPanel	-	XYData replication	replication – результати реплікації даних
ReplicationPanel	initialize	Ініціалізація панелі виводу результатів реплікації	-	-	-
Simulation	init	Сконфігурувати сервери та дані	-	-	-
Simulation	run	Запустити симуляцію системи	-	-	-

Продовження таблиці 3.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Simulation	stop	Зупинити роботу системи	-	-	-
Simulation	save	Зберегти дані симуляції			
Simulation	stop	Зупинити роботу серверу	-	String serverId	serverId – номер серверу
Generate	generateVM Character	Сформувати ВМ		Server server	server – сервер для розміщення ВМ
Device	addFile	Додати файл до дисковий пристрій	Boolean	File file	file – файл для запису
Device	searchFile	Знайти файл на дисковому пристрої	File	File file	file – файл для пошуку
Device	checkFile	Перевірити наявність файлу на дисковому пристрої	Boolean	File file	file – файл для перевірки
Device	removeFile	Видалити файл з дискового пристрою	-	File file	file – файл для видалення
Device	renewFile	Обновити файл на дисковому пристрої	Boolean	File file	file – оновлений файл
Device	makeCopy	Створити копію дискового пристрою	List<File>	-	-
Device	cleanAccess Block	Очистити доступ до блоків	-	-	-
Device	increaseAcce ssFile	Створити копію дискового пристрою	-	File file	file – файл
Storage	addFasterDe vice	Додати пристрій до більш швидкого рівня	-	Disk disk	disk – пристрій
Storage	addSlowerDe vice	Додати пристрій до більш повільного рівня	-	Disk disk	disk – пристрій
Storage	addFile	Додати файл до сховища	-	File file	file – файл

Продовження таблиці 3.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Storage	renewFile	Обновити файл на сховищі	-	File file	file – файл
Storage	readFile	Прочитати файл зі сховища	File	File file	file – файл
Storage	removeFile	Видалити файл зі сховища	-	File file	file – файл
Storage	cleanAccess	Очистити доступ до даних сховища	-	-	-
Storage	getTotalSize FasterTier	Отримати загальний розмір пристроїв на першому рівні	Double	-	-
Storage	getTotalSize SlowerTier	Отримати загальний розмір пристроїв на другому рівні	Double	-	-
Storage	getNonFreeS pace	Отримати розмір вільного місця на пристроях другого рівню	Double	-	-
Storage	migration	Міграція даних між рівнями	-	-	-
Storage	madeReplica tion	Створити реплікацію даних	-	-	-
VM	Run	Запустити ВМ на сервері			
Server	addVM	Додати ВМ до серверу	-	VM vm	vm – ВМ
Server	addSlower	Додати пристрій на другий рівень	-	Disk disk	disk – дисковий пристрій
Server	addFaster	Додати пристрій на перший рівень	-	Disk disk	disk – дисковий пристрій
Server	addReplica	Додати копії даних з інших серверів	-	List<File> replicaData	replicaData -файли для збереженн я
Server	totalUsage	Перевірка наявності пропускної спроможності на сервері	boolean	-	-

Продовження таблиці 3.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Server	Run	Запустити сервер	-	-	-
Server	getCountExecuteProcesses	Додати кількість процесів запущені на сховищі	Integer	-	-
Replication	makeReplication	Додати копії даних з інших серверів	-	Disk disk	disk – дисковий пристрій
Migration	setFreeSpace	Позначити необхідну кількість вільного місця на першому рівні сховища	-	Double freeSpace	freeSpace – кількість вільного місця
Migration	migration	Мігрувати дані з другого рівня на перший рівень	boolean	File file	file - файл
Migration	fillFreeSpace	Звільнити місце на першому рівні	-	-	-
Sort	sort	Відсортувати список за двома критеріями	List<T>	List<T> list, Double weightFirst, Double weightSecond, Comparator<T> sortFirst, Comparator<T> sortSecond	List-список об’єктів для сортування, weightFirst – вага першого критерію, weightSecond – вага другого критерію, sortFirst – сортування списку за першим критерієм, sortSecond – сортування списку за другим критерієм

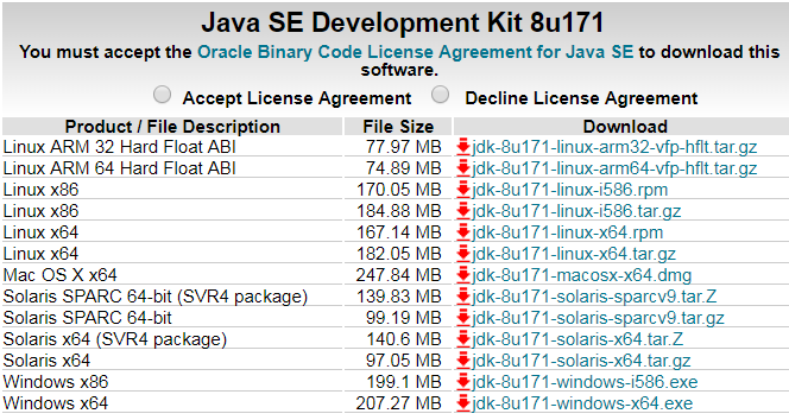
3.2.4 Керівництво користувача

Для коректної роботи розробленого програмного продукту, в ОС повина бути налаштована JAVA.

Для її налаштування необхідно перейти за посиланням http://www.oracle.com/technetwork/java/javase/downloads/index.html, обрати останню або необхідну користувачу версію JAVA (бажано 8 або вище) та в обраній версії натиснути на посилання «Download» (рисунок 3.8), яке відповідає позначкам JRE або JDK, в залежності від мети налаштування JAVA.

Рисунок 3.8 – Вибір способу налаштування JAVA на комп’ютер користувача програмним продуктом

На сторінці, яка буде відкрита необхідно обрати ОС, яка налаштована на даному ПК, погодитись з умовами користування (рисунок 3.9) та завантажити необхідну версію JAVA з розширенням .exe на комп`ютер користувача, який на даний момент користується ОС – Windows або файл з розширенням tar.gz/rpm – при запуску програмного продукту на ОС Linux .



Java SE Development Kit 8u171

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.97 MB	jdk-8u171-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.89 MB	jdk-8u171-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.05 MB	jdk-8u171-linux-i586.rpm
Linux x86	184.88 MB	jdk-8u171-linux-i586.tar.gz
Linux x64	167.14 MB	jdk-8u171-linux-x64.rpm
Linux x64	182.05 MB	jdk-8u171-linux-x64.tar.gz
Mac OS X x64	247.84 MB	jdk-8u171-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.83 MB	jdk-8u171-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.19 MB	jdk-8u171-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.6 MB	jdk-8u171-solaris-x64.tar.Z
Solaris x64	97.05 MB	jdk-8u171-solaris-x64.tar.gz
Windows x86	199.1 MB	jdk-8u171-windows-i586.exe
Windows x64	207.27 MB	jdk-8u171-windows-x64.exe

Рисунок 3.9 – Сторінка вибору ОС для налаштування JAVA

Після закінчення завантаження, необхідно відкрити збережений файл (рисунок 3.10) та натиснути на кнопку «Install».

Рисунок 3.10 – Форма початку налаштування JAVA

В результаті відкриється форма налаштування JAVA та почнеться процес налаштування на поточний комп’ютер з ОС – Windows (рисунок 3.11).

Рисунок 3.11 – Форма налаштування JAVA

Після закінчення налаштування відкриється форма завершення як показано на рисунку 3.12.

Рисунок 3.12 – Завершення налаштування JAVA на ОС

Для запуску програми необхідно розпакувати архів master.zip та відкрити файл запуску програми з розширенням .jar. Після чого відкриється форма для введення даних для роботи сервісу (рисунок 3.13).

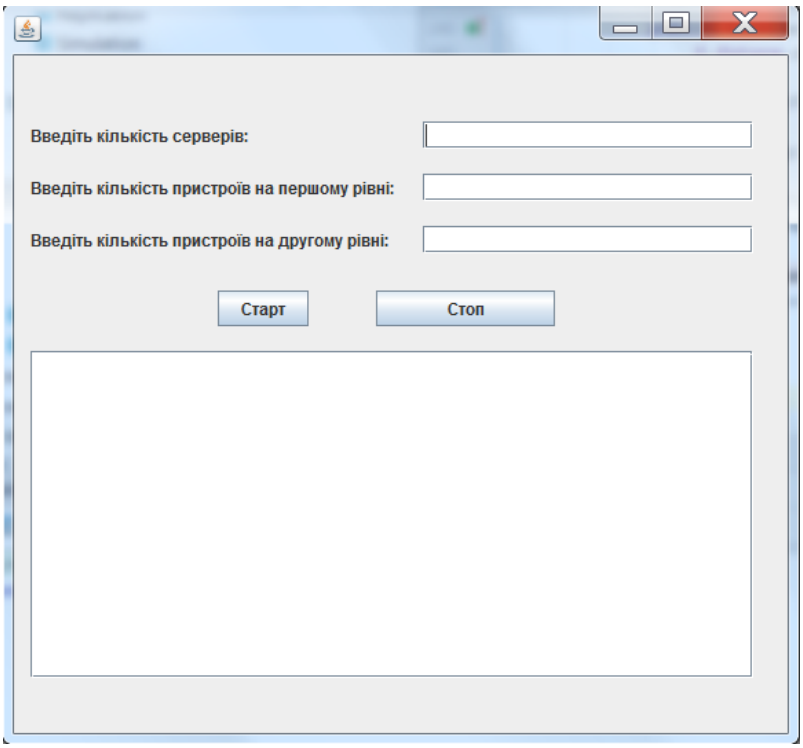


Рисунок 3.13 – Введення кількості серверів та пристроїв на рівнях необхідної для симуляції роботи сховища

Користувачу необхідно ввести:

- кількість серверів, які необхідні для симуляції;
- кількість пристроїв, які знаходяться на першому рівні на кожному з серверів системи;
- кількість пристроїв, які знаходяться на другому рівні, але слід зауважити, що їх продуктивність є гіршою, порівняно з пристроями першого рівня кожного серверу.

Після введення необхідно натиснути на кнопку «Старт», яка знаходиться на представленій формі, що в результаті розпочне роботу симулятор системи сховища в гіперконвергентній системі (рисунок 3.14).

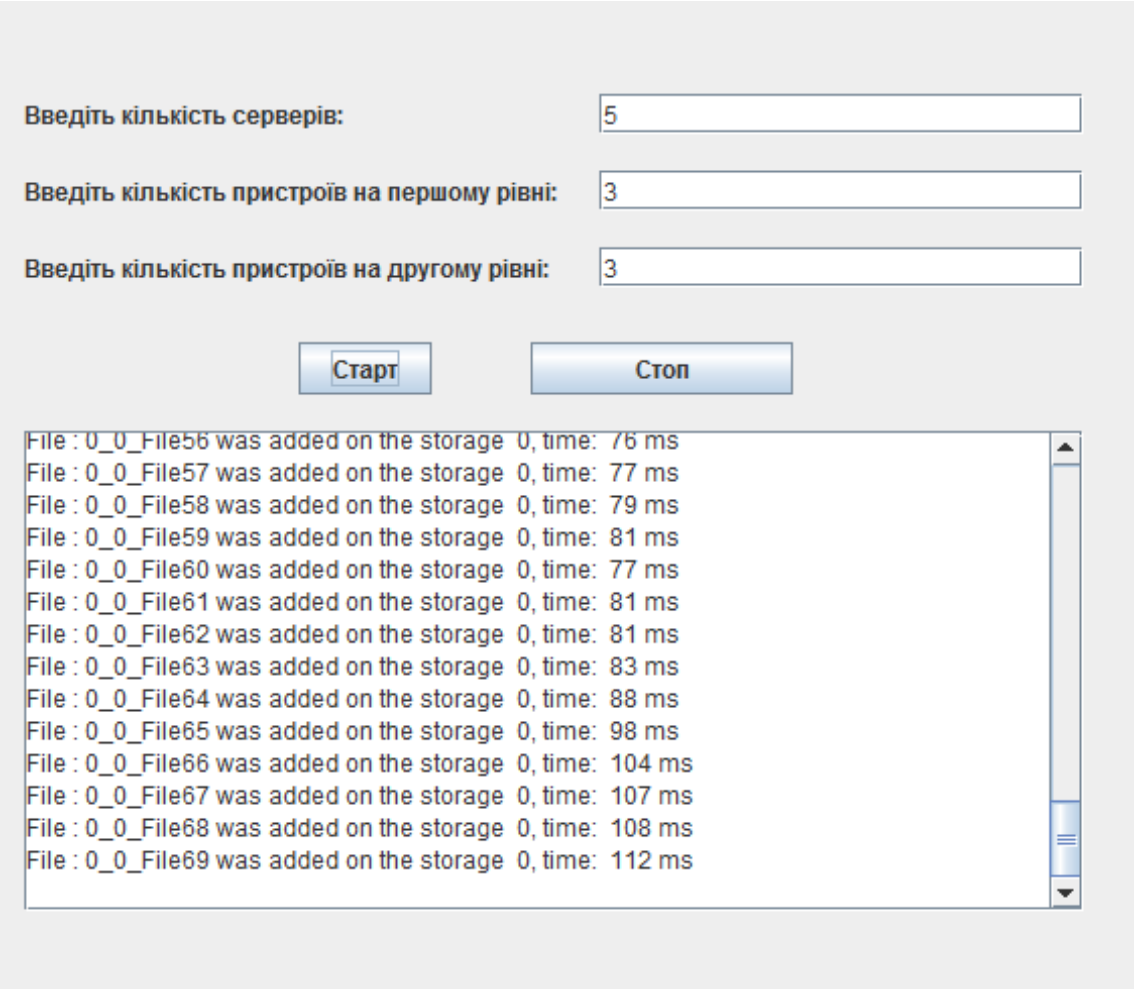


Рисунок 3.14 - Симулятор роботи сховища в гіперконвергентній системі

Симулятор створює введену кількість серверів в системі. На кожному створеному сервері створюються два рівні пристроїв, кількість яких було введено користувачем в відповідних полях «Введіть кількість пристроїв на першому рівні» та «Введіть кількість пристроїв на другому рівні» у формі програмного продукту. Після створення серверів в системі, починають генеруватись віртуальні машини на сховищі. Дані для віртуальних машин беруться з файлів, які були сформовані за допомогою звітів, які створюються при роботі з програмним застосуванням Process Monitor (рисунок 3.15).

2	"C:\Program Files\Common Files\McAfee\platform\McSvcHost\McSvHost.exe"	596768	FALSE	276
3	"C:\Windows\System32\adtschema.dll"	690688	FALSE	64
4	"C:\Windows\System32\apisetschema.dll"	6656	FALSE	70
5	"C:\Windows\System32\ntdll.dll"	1665384	FALSE	39
6	"C:\Windows\System32\kernel32.dll"	1163264	FALSE	51
7	"C:\Windows\System32\KernelBase.dll"	419840	FALSE	41
8	"C:\Windows\System32\locale.nls"	419648	FALSE	39
9	"C:\Windows\System32\svchost.exe"	27136	FALSE	542
10	"C:\Windows\System32\msvcrt.dll"	634880	FALSE	41
11	"C:\Windows\System32\sechost.dll"	113664	FALSE	121
12	"C:\Windows\System32\rpcrt4.dll"	1212928	FALSE	41
13	"C:\Windows\Globalization\Sorting\SortDefault.nls"	2944004	FALSE	78
14	"C:\Windows\System32\wersvc.dll"	76800	FALSE	78
15	"C:\Windows\System32\advapi32.dll"	880640	FALSE	537
16	"C:\Windows\System32\wlansvc.dll"	886272	FALSE	8
17	"C:\Windows\System32\services.exe"	328704	FALSE	4
18	"C:\Windows\System32\dhcpcore.dll"	317952	FALSE	1

Рисунок 3.15 – Приклад даних, які подаються на вхід програмного продукту для симуляції роботи ВМ

Дані для збереження на сховищі починають зберігатись на сховище серверу на якому знаходиться ВМ. Після закінчення налаштування серверів, від кожної ВМ починають приходити запити до сховища для читання або запису даних. Кількість запитів для обробки кожної ВМ зберігаються в кожному файлі з вхідними даними.

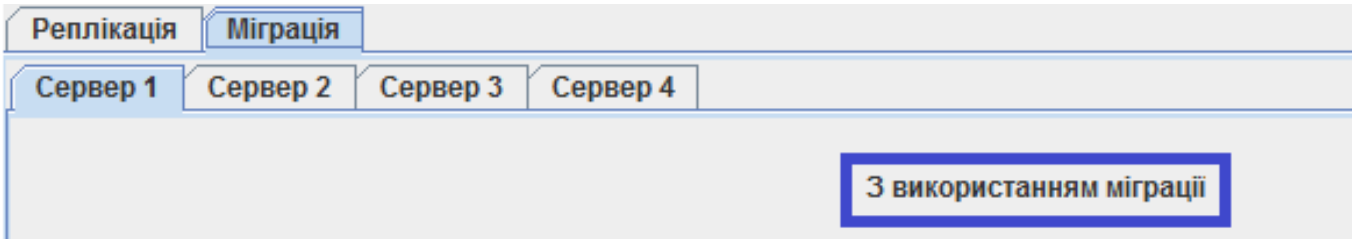
В момент, коли користувач має намір отримати результати симуляції роботи сховища, стає необхідним зупинки процесу симуляції. Для цього необхідно на формі натиснути на кнопку «Стоп», після чого відкриється форма результатів роботи системи сховищ.

На формі користувач має можливість побачити результати роботи реплікації та міграції в системі. Для цього йому необхідно переходити за посиланнями «Міграція» або «Реплікація» на формі (рисунок 3.16) в залежності від результатів, які йому необхідні на даний момент.

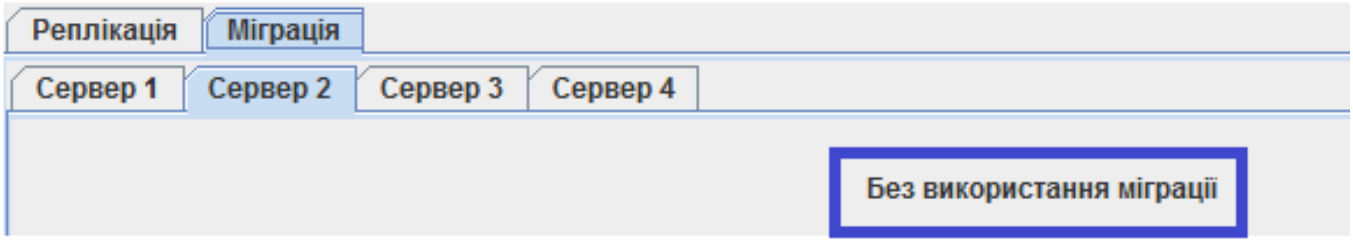
Рисунок 3.16 – Вибір задач, результати яких необхідні

При виборі панелі «Міграція» користувачу надається можливість вибору результатів роботи кожного серверу окремо. На серверах, номер яких є непарним міграція присутня, на інших вона відсутня. Побачити, чи відбувалась міграція на

поточному сервері можна за допомогою позначки на панелі як показано на рисунку 3.17а – 3.17б.



а)



б)

Рисунок 3.17 – Перевірка наявності міграції на поточному сервері програмного продукту

Також на панелі міграції можна побачити результати роботи програмного продукту, які складаються з відображення даних у вигляді трьох графіків. Перший графік відображає залежність часу роботи системи від кількості міграцій, які відбулись на ній (рисунок 3.18). У випадку вибору серверів, міграція даних на яких міграція була відсутня протягом часу симуляції поверне даний графік порожнім (рисунок 3. 19) .

Рисунок 3.18 – Результати виконання міграції (частина 1)

Рисунок 3.19 – Результат роботи серверу без міграції

Наступний графік представлений на панелі міграції кожного серверу показує результати читання даних з сховищ серверу системи віртуальними машинами (рисунок 3.20).

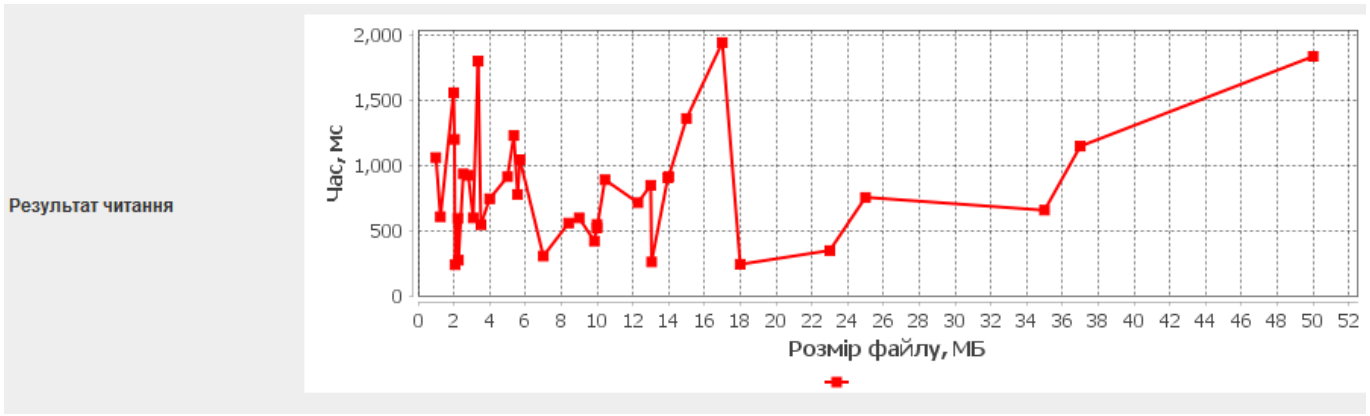


Рисунок 3.20 – Результати виконання міграції (частина 2)

Останній графік на панелях міграції містить інформацію про час запису файлів, які належать віртуальним машинам до сховищ серверів системи в залежності від їх розмірів (рисунок 3.21).

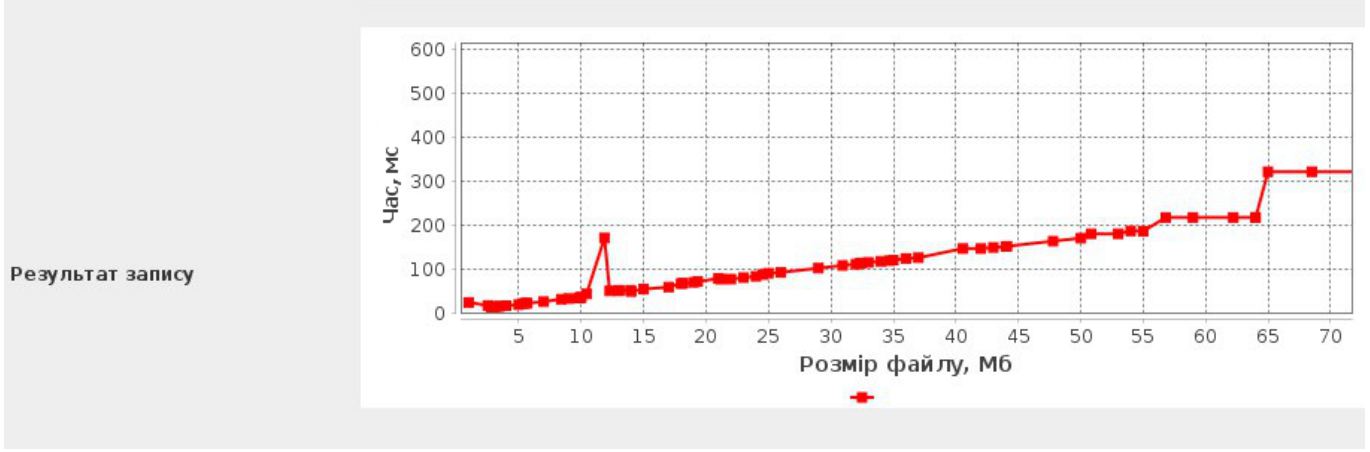


Рисунок 3.21 – Результати виконання міграції (частина 3)

Також при аналізі міграції можна побачити значення кількості вільного місця, яке зараз є на першому рівні сховища (рисунок 3.22).

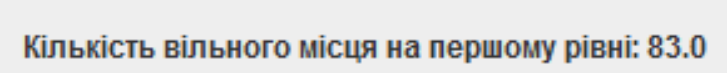


Рисунок 3.22 – Результати виконання міграції (частина 4)

Для перегляду результатів міграції необхідно перейти в директорію «Result» та відкрити CSV файл з результатами міграції файлів під час моделювання (рисунок 3.23).

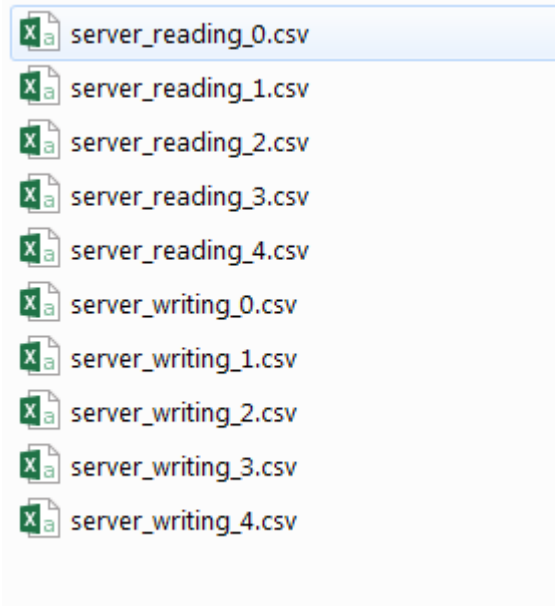


Рисунок 3.23- Збережені результати міграції даних

Приклад результатів представлено на рисунку 3.24.

Розмір файлу,МБ	Час, мс
1	556.25
1.253417969	817.875
1.986026764	742.4375
2.035644531	639.71875
2.064094543	266
2.22530365	646.5
2.255859375	1428.5
2.529296875	186.8125
2.807621002	619.625
3.080078125	1033
3.354064941	1643.65625
3.512046814	381.5
4	214.25
5	809.1582031
5.3515625	1059.265625
5.5625	1319.636963

Рисунок 3.24 – Результати читання/запису файлів під час моделювання роботи сховища

У випадку отримання результатів реплікації даних між декількома фізичними серверами системи, користувачу необхідно обрати на формі посилання «Реплікація», після чого можна побачити результати моделювання роботи гіперконвергентної системи. Результати представлені у вигляді двох таблиць:

− розподілення реплікацій даних між фізичними серверами системи моделювання (рисунок 3.25);

− поточна завантаженість дискових пристроїв даними з поточного фізичного серверу та реплікаціями (рисунок 3.26).

Розподілення реплікацій

	Сервер 1	Сервер 2	Сервер 3	Сервер 4	Сервер 5
Сервер 1		0	0	0	0
Сервер 2	0		0	0	0
Сервер 3	0	0		0	0
Сервер 4	0	0	0		0
Сервер 5	0	0	0	0	

Рисунок 3.25 – Результати виконання реплікації (частина 1)

З результатами щодо загального розміру другого рівня серверів, поточної заповненості пристроїв даними та об’ємом реплікацій даних з інших серверів, які зберігаються на кожному сервері можна ознайомитись за допомогою другої таблиці, яка представлена на формі.

Завантаженість дисків

	Загальний розмір сховища, МБ	Зайнятість сховища даними, МБ	Зайнятість сховища реплікація...
Сервер 1	50000.0	0.0	0.0
Сервер 2	50000.0	0.0	0.0
Сервер 3	50000.0	0.0	0.0
Сервер 4	50000.0	0.0	0.0
Сервер 5	50000.0	0.0	0.0

Рисунок 3.26 – Результати виконання реплікації (частина 2)

3.3 Висновок до розділу

В розділі «Опис розробленого програмного забезпечення» було представлено інформаційне забезпечення розробленого програмного продукту, до якого входить процес формування вхідних та вихідних даних. Представлено опис програмного та технічного забезпечення програмного продукту, який складався з засобів розробки,

вимог до технічного забезпечення, діаграми класів, специфікації функцій та керівництво користувача для роботи з розробленим програмним продуктом магістерскої дисертації.

Також надано діаграму класів та специфікацію функцій додаткового застосування, який створено для формування вхідних даних для роботи з програмним продуктом та за допомогою якого було проаналізовано необхідність використання застовування Manager Monitor для аналізу використання дискового пристрою процесами, яким необхідно виконати операції читання та запису при роботі ОС Windows.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Порядок проведення досліджень

4.1.1 Процес формування вхідних даних

На початку проведення досліджень сформовано вхідні дані з використанням програмного застосування Process Monitor. Збір вхідних даних виконано в декілька етапів, які описані в пункті 3.1.1.

Перший етап. Збір даних про використання дискового пристрою користувачем відбувався з використанням застосування Process Monitor протягом деякого часу при роботі з операційною системою Windows.

Другий етап. Формування звіту застосуванням Process Monitor у форматі CSV, в який записуються результати, отримані програмою протягом часу, за який відбувався збір даних на комп'ютері.

Третій етап. Формування вхідних даних для дослідження якості використання представлених в роботі алгоритмів. Для виконання даного етапу дослідження розроблене окреме програмне застосування, яке описано в пункті 3.2.2.1 за допомогою діаграми класів.

Отримані результати використані при роботі з розробленим програмним продуктом.

Також, вхідними даними для аналізу роботи алгоритмів є дані, які були отримані від користувача системи. До цих даних відносяться:

- кількість серверів в кластері;
- кількість дискових пристроїв, які знаходяться на першому рівні сховища (мають вищі показники продуктивності);
- кількість дискових пристроїв з нижчими показниками продуктивності, які знаходяться на другому рівні сховищ даних;
- кількість часу роботи системи.

Під час моделювання, для серверних систем створювались два типи сховищ: багаторівневі та однорівневі. Однорівневі мають тільки один рівень пристроїв зберігання даних з нижчими показниками продуктивності. Пристрої, які розміщувались на баготорівневих сховищах поділялись в залежності від характеристик, таких як затримка в очікувані відповіді від пристрою, середній час пошуку (доступу) до файлів та пропускна спроможність дискових пристроїв.

Характеристики роботи дискових пристроїв багаторівневих сховищ, які розміщуються на першому рівні, є наступними:

- затримка – 1 мс;
- середній час доступу до файлів – 1 мс;
- пропускна спроможність дискового пристрою – 300 МБ/сек.

Характеристики роботи дискових пристроїв на другому рівні багаторівневих сховищ:

- затримка – 6 мс;
- середній час доступу до файлів – 9 мс;
- пропускна спроможність дискового пристрою – 120 МБ/сек.

Характеристики пристроїв, які належать серверам, на яких налаштовані однорівневі сховища, є наступними:

- затримка – 3 мс;
- середній час доступу до файлів – 6 мс;
- пропускна спроможність дискового пристрою – 150 МБ/сек.

4.1.2 Результати міграції даних

Для отримання даних при роботі з системою моделювання використано CSV файл, який зберігає дані про доступ до файлів ОС Windows загальним розміром 9700 Мб. Міграція даних між рівнями сховища одного фізичного серверу відбувалась у випадку наявності певної визначеної кількості доступів до файлу ОС, що приводило до необхідності міграції. При великому навантажені на сервер, коли постійно, протягом всього часу моделювання, від кожної віртуальної машини до сховища

надходять запити на обробку великої кількості даних, міграція допомогає зменшити час очікування даних зі сховищ шляхом збереження часто запитуваних файлів на пристроях, продуктивність яких є вищою порівняно з іншими дисковими пристроями сховища.

На рисунку 4.1 можна побачити, що після початку роботи системи, деякий проміжок часу під час її роботи, необхідність міграції даних між рівнями сховища відсутня через те, що відбувалась постійна необхідність використання сховища тільки для виконання операцій записів на дискові пристрої. В момент, коли почалось постійне використання сховищ даних для виконання операцій запису і читання зі сховища, кількість міграцій між рівнями сховища даних почала помітно збільшуватись через перенесення файлів, доступ до яких почав відбуватись часто, на більш швидкий рівень з менш повільного рівня за рахунок звільнення місця на дискових пристроях першшого рівня.

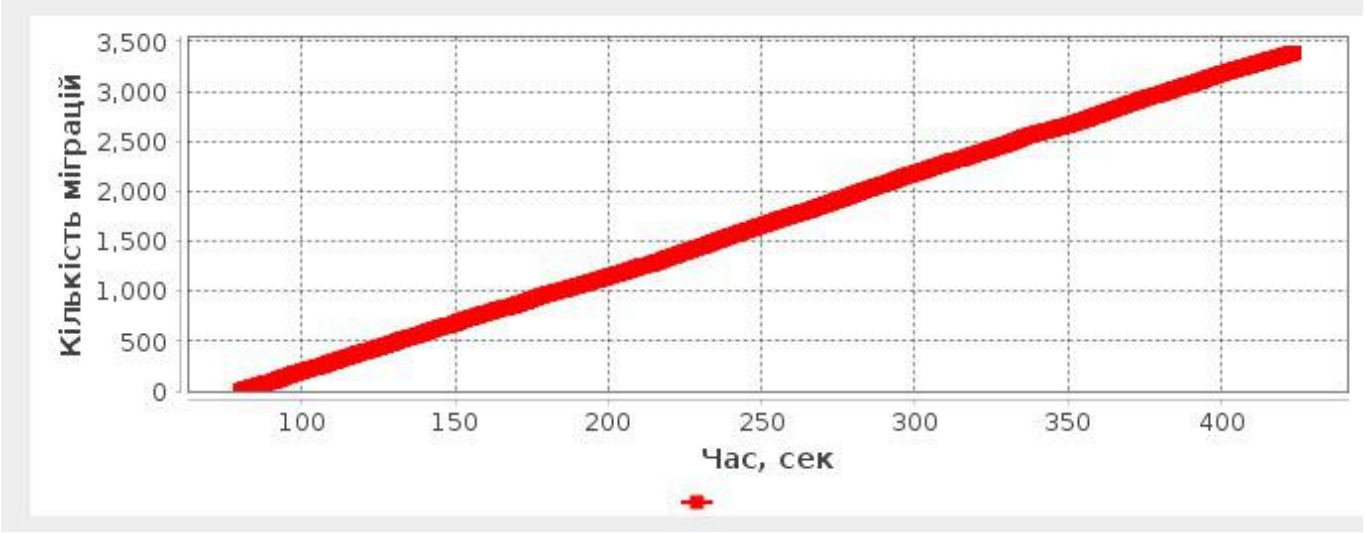


Рисунок 4.1 – Кількість міграцій файлів на одному фізичному сервері в системі моделювання

В таблицях 4.1-4.2 представлена частина результатів, отриманих при моделювані операцій записів файлів на багаторівневі та однорівневі сховища фізичних серверів.

фізичних серверів.

Таблиця 4.1 – Результати додавання файлів до серверів з наявною міграцією (часткові)

Розмір файла, МБ	Час запису, мс
8.4101187	34
9.999645	36
12.28369	83
17	58
18.15625	94
19.3291	96
24.54785	104
30.93319	116
34.69733	119
44.09795	159
47.77631	165
54	184
64	216

Таблиця 4.2 – Результати додавання файлів до серверів з відсутньою міграцією (часткові)

Розмір файла, МБ	Час запису, мс
8.4101187	62
9.999645	69
12.28369	167
17	124
18.15625	249
19.3291	249
24.54785	249
30.93319	249
34.69733	239
44.09795	337
47.77631	337
54	363
64	429

Так як запис файлів при використані міграції постійно відбувається на перший рівень, то час запису файлів, у випадку відсутності місця на першому рівні буде компенсуватись за рахунок видалення файлів з першого рівня сховища. При цьому видаляються файли з нижчими показниками розміру та кількості відвідуваності. Якщо розглядати ці результати графічно (рисунок 4.2 – 4.3), то можна побачити, що час запису через різні характеритистики пристроїв відрізняється, але так як запис на багаторівневі сховища відбувається постійно на перший рівень, то результати очікування виконання операцій запису на них є набагато кращі, ніж на сховища, в яких відсутня міграція та налаштування декількох рівнів з пристроями, які мають різні характеристики. В момент, коли вільне місце на серверах без міграції закінчується, час запису на сервери збільшується через те, що характеристики пристроїв на другому рівні є гіршими.

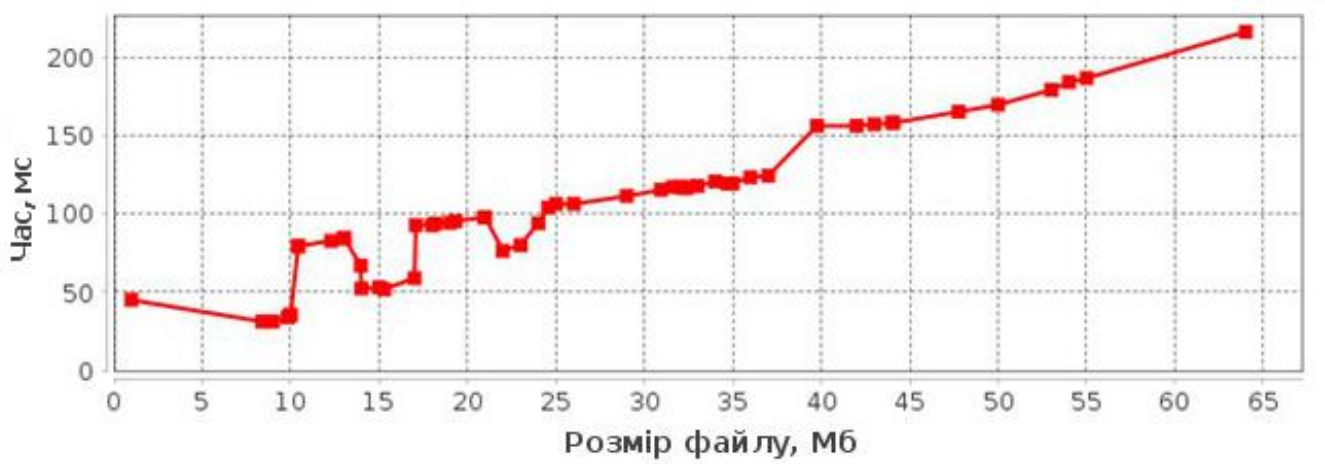


Рисунок 4.2 – Час запису файлів в залежності від їх розміру з наявною міграцією

Рисунок 4.3 – Час запису файлів в залежності від їх розміру з відсутньою міграцією

Під час аналізу роботи сховища в процесі читання файлів при використані міграції, можна побачити, що при постійному доступу до файлу з наявною міграцією на сховищах, час очікування доступу до файлу є меншим через те, що дані знаходяться на першому рівні.

В таблицях 4.3 – 4.4 представлені часткові результати часу читання даних з багаторівневих сховищ та сховищ, які мають тільки один рівень пристроїв для збереження відповідно.

Таблиця 4.3 – Результати додавання файлів до сховища з наявною міграцією (часткові)

Розмір файла, МБ	Час запису, мс
4	423
5	187
8.410187	267
10.43871	199
12.28369	420
13	320
18	297
23	709
25	320

Продовження таблиці 4.3

35	315
50	550

Таблиця 4.2 – Результати додавання файлів до сховища з відсутньою міграцією(часткові)

Розмір файла, МБ	Час запису, мс
4	300
5	187
8.410187	245
10.43871	170
12.28369	410
13	340
18	499
23	600
25	343
35	475
37	480
50	565

Розглядаючи повні результати дослідження графічно, які представлені на рисунках 4.4 - 4.5, можна побачити необхідність використання алгоритму міграції для зменшення часу очікування віртуальними машинами при постійній навантаженості серверу запитами на читання даних.

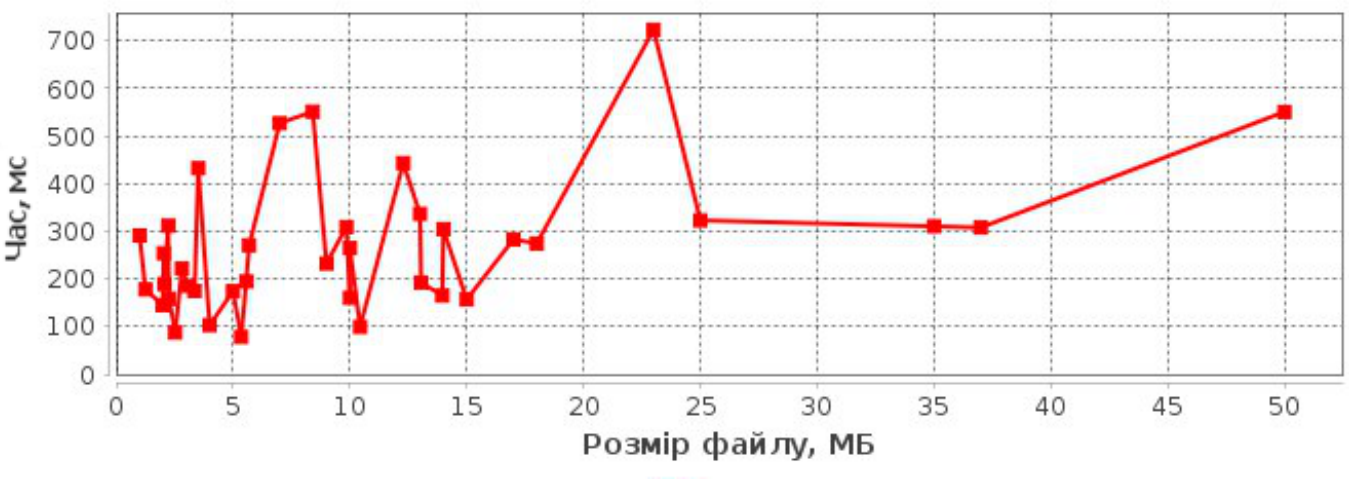


Рисунок 4.4 – Результат читання файлів в залежності від їх розміру з наявною міграцією

Рисунок 4.5 – Результат читання файлів в залежності від їх розміру без міграції

Аналізуючи отримані результати можна побачити, що у більшості випадків час запису даних на пристрої є меншим у випадку використання міграції між рівнями сховища. У випадках, коли файли зберігаються на другому рівні сховища, час читання файлів зі сховищ з наявною міграцією перевищує час читання файлів зі сховищ, да міграція відсутня. Якщо розглядати усереднені значення запису деякої кількості файлів з використанням міграції та без неї, то використання міграції допомогає зменшити час очікування файлів при виконання операцій читання віртуальними машинами на серверах.

Отримані результати дослідження роботи системи моделювання показують, що налаштування багаторівневих сховищ з алгоритмами управління процесами збереження даних на них, в залежності від критеріїв кількості доступу та розміру

файлів, призводить до зменшення часу очікування завершення операцій читання та запису при роботі зі сховищами віртуальними машинами. Час запису з використанням представленого алгоритму та налаштуванням багаторівневих сховищ є меншим порівняно зі сховищами, в яких відсутні рівні та міграція. Аналізуючи час читання файлів зі сховищ, то усереднені значення часу очікування є меншими у випадку налаштування багаторівневих сховищ, через те, що файли, доступ до яких відбувається постійно, знаходяться на першому рівні, характеристики пристроїв якого є кращими.

4.1.3 Результати реплікації даних

Для дослідження уникнення втрати даних на сховищі стає необхідним використовувати механізм реплікації даних. Так як налаштовувалось гіперконвергентне сховище з декількома серверами, які складаються зі сховищ та віртуальних машин, реплікація стає необхідною не тільки для збереження даних на одному (оригінал) сервері, а й для використання інших серверів.

Вибір серверів-реплік повинен відбуватись в залежності від поточної їх завантаженості. Завантаженість серверу залежить від двох критеріїв:

− наявність вільного місця на другому рівні сховища, необхідного для зберігання даних;

− якомога менше реплікацій даних з інших серверів дані яких зберігаються на поточному сервері.

За допомогою реплікації даних створюються дві копії кожного дискового пристрою сховища, які розміщюються на серверах, показники продуктивності описаних вище критеріїв яких є краще. Створення двух копій дозволяє завантажувати всі сервери рівномірно, для уникнення їх перевантаження

При проведені дослідження було створено центр обробки даних, який складався з п’яти фізичних серверів на кожному з яких знаходилось сховище даних.

Результати дослідження представлені на рисунках 4.6 – 4.7.

	Сервер 1	Сервер 2	Сервер 3	Сервер 4	Сервер 5
Сервер 1		1	0	1	0
Сервер 2	0		1	0	1
Сервер 3	1	0		1	0
Сервер 4	0	0	1		1
Сервер 5	1	1	0	0	

Рисунок 4.6 – Розподілення реплікацій даних між серверами під час моделювання системи(рядки – сервери, дані з яких реплікуються; стовпці – сервери на які реплікують дані)

	Загальний розмір сховища, МБ	Зайнятість сховища даними, МБ	Зайнятість сховища реплікація...
Сервер 1	50000.0	35415.0	23040.0
Сервер 2	50000.0	34319.0	23802.0
Сервер 3	50000.0	31349.0	19736.0
Сервер 4	50000.0	33207.0	23988.0
Сервер 5	50000.0	31163.0	19736.0

Рисунок 4.7 – Зайнятість сховищ даними разом з реплікаціями

Для перевірки достовірності роботи представленого алгоритму можна розрахувати кількість даних, не враховуючи реплікації, які зберігаються на кожному фізичному сервері сховища (таблиця 4.5).

Таблиця 4.5 – Зайнятість сховищ фізичних серверів даними (етап 1)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувалась реплікація
1	12375	0	-
2	10517	0	-
3	11613	0	-
4	9219	0	-
5	11427	0	-

Першим кроком алгоритму є пошук місця для реплікацій даних з першого серверу. Так як реплікації на даному етапі ще не були виконані з інших серверів, то необхідно обирати серверів на яких більша кількість вільного місця. Аналізуючи поточну завантаженість серверів, то це сервери 2 та 4. Результати виконання другого етапу реплікації наведені в таблиці 4.6.

Таблиця 4.6 – Зайнятість сховищ фізичних серверів даними (етап 2)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувається міграція
1	12375	0	2,4
2	22892	1	-
3	11613	0	-
4	21594	1	-
5	11427	0	-

Наступним кроком роботи системи є пошук місця для реплікацій даних для другого серверу. Розглядаючи поточні показники серверів, реплікації даних необхідно робити на третій та четвертий сервери системи моделювання, так як на поточний момент кількість вільного місця на них є найбільшою та на них не відбувається реплікація даних з інших серверів. Результати виконання наступного етапу реплікаці даних між фізичними серверами системи моделювання представлені в таблиці 4.7.

Таблиця 4.7 – Зайнятість сховищ фізичних серверів даними (етап 3)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувалась
1	12375	0	2,4
2	22892	1	3,5
3	22130	1	-
4	21594	1	-
5	21944	1	-

Для пошуку місця для збереження реплікацій даних третього серверу необхідно обрати сервери показники зайнятості та кількості реплікацій яких є найменшими. Опираючись на поточні показники серверів, для збереження реплік потрібно обрати

сервери 1 та 4. Перший сервер обирається через те, що на поточний момент не було виконано реплікацій на нього з інших серверів системи та його зайнятість даними на даний момент є найменшою. Результати виконання етапу 4 представлені в таблиці 4.8.

Таблиця 4.8 – Зайнятість сховищ фізичних серверів даними (етап 4)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувається реплікація
1	23988	1	2,4
2	22892	1	3,5
3	22130	1	1,4
4	33207	2	-
5	21944	1	-

Наступним етапом є пошук місця для розміщення реплік даних з четвертого серверу. Так як кількість реплікацій на інших серверах на даний момент дорівнює одиниці, то сервери для розміщення реплікацій обираються в залежності від кількості вільного місця на серверах. Опираючись на це, серверами для збереження реплікацій даних з поточного серверу є 3 та 5 сервери. Результати виконання етапу представлені в таблиці 4.9.

Таблиця 4.9 – Зайнятість сховищ фізичних серверів даними (етап 5)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувається реплікація
1	23988	1	2,4
2	22892	1	3,5
3	31349	2	1,4
4	33207	2	3,5
5	31163	2	-

Для пошуку місця для збереження реплік даних для серверу 5 необхідно обрати найменш завантажені на поточний момент сервери. Аналізуючи поточний стан кожного серверу, найменш завантаженими, як видно з таблиці 4.9, є сервери 1 та 2, так як на поточний момент сервери зберігають по одному екземпляру копій даних з інших серверів та кількість вільного місця на них є нижчою порівняно з іншими

фізичними серверами системи. Результати викориння етапу 6 представлені в таблиці 4.10.

Таблиця 4.10 – Зайнятість сховищ фізичних серверів даними (етап 6)

№ серверу	Кількість зайнятого місця на сховищі, МБ	Кількість реплікацій	№ серверів, на які відбувається реплікація
1	35415	2	2,4
2	34319	2	3,5
3	31349	2	1,4
4	33207	2	3,5
5	31163	2	1,2

Порівнюючи з результатами, які повернула система моделювання, можна побачити коректність роботи алгоритму. Розподілення між фізичними серверами системи відбулось рівномірно.

4.2 Висновки до розділу

В представленому розділі проаналізовано отримані результати використання алгоритмів міграції та реплікації даних для управління процесами запису/читання зі сховищ центрів обробки даних. Також, виконано порівняння роботи сховищ даних з використанням представлених алгоритмів та не використовуючи їх. Описано процес формування вхідних та вихідних даних, які були отримані при моделювані системи за допомогою CSV файлів, які зберігають інформацію про результати роботи розробленого програмного продукту при виконані магістерскої дисертації.

ЗАГАЛЬНІ ВИСНОВКИ

При виконані магістерської дисертації проаналізовано предметне середовище організації сховищ даних в гіперконвергентних системах. Представлені в роботі математичні моделі використовують наступні критерії при оптимізації роботи зі сховищами даних:

- зменшення часу очікування виконання операцій читання і запису зі сховища;
- уникнення втрати даних з фізичних серверів.

Для кожної з моделей представлено алгоритми, які в залежності від поточних станів компонентів системи, обирають місця розміщення реплік файлів на сховищах фізичних серверів.

Представлений алгоритм для вирішення проблеми зменшення часу виконання операцій читання і запису, що використовує міграцію файлів між двома рівнями сховища в залежності від розміру файлів та кількості разів доступу до них протягом заданого часу. Аналіз результатів, отриманих при роботі розробленого алгоритму, показав, що при виконані операцій запису файлів до сховища розроблений алгоритм деионструє кращі показники ефективності порівняно з результатами сховищ, які не налаштовані на декілька рівнів збереження. При аналізі отриманих результатів виконання операцій читання файлів зі сховища, можна побачити, що не завжди час читання файлів є меншим порівняно зі сховищами без міграції, але в усередненому часі виконання операцій читання використання алгоритму міграції даних показує кращі результати порівняно зі сховищами, на яких міграція даних відсутня.

Способом уникнення втрати інформації на фізичних серверах є налаштування реплікації копій даних між серверами. Розроблений алгоритм для реплікації даних полягає у виборі фізичних серверів системи для збереження копій даних в залежності від поточних показників завантаженості серверів. Для порівняння серверів з метою розміщення реплік вводяться коєфіцієнти завантаженості фізичних серверів, які відображають кількість вільного місця на кожному фізичному сервері та кількість реплікацій даних з інших серверів, які на даний момент зберігають дані на поточному сервері. Результати використання представленого алгоритму показали, що

розміщення реплікацій даних на фізичних серверах відбувається рівномірно та коєфіцієнти завантаженості серверів є рівномірними по критеріям кількості вільного місця на фізичному сервері та кількості реплікацій.

Отже, результати, які отримані при досліджені роботи алгоритмів міграції та реплікації даних на сховищах, показали зменшення часу очікування на обробку операцій читання/запису файлів та рівномірну завантаженість фізичних серверів даними та реплікаціями даних з інших серверів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Сягайло Т.А., Жаріков Е.В. Порівняльний аналіз алгоритмів для управління розміщенням віртуальних машин. / Сягайло Т.А., Жаріков Е.В. // Всеукраїнська науково практичної конференції «Комп'ютерні інтелектуальні системи та мережі». - 2017. – СС.10 – 13.

2. Сягайло Т.А. Способи налаштування сховищ в гіперконвергентних системах./ Жаріков Е.В., Сягайло Т.А., Коваль А.А. // Міжнародна наукова конференція «Актуальні наукові дослідження в сучасному світі». - 2018. - СС.41 - 51.

3. Сягайло Т.А., Жаріков Е.В. Управління процесами збереження даних в гіперконвергентних системах/ Жаріков Е.В., Сягайло Т.А. // Наукова конференція IOT-2018 (АСОІУ). – 2018.

4. Терентьєв Р.А., Сягайло Т.А. Вплив розміру тренувальних даних на точність прогнозу потреби ресурсів серверних систем в умовах хмарних обчислень / Терентьєв Р.А., Сягайло Т.А. // Міжнародна наукова конференція «Актуальні наукові дослідження в сучасному світі». - 2018. - С.108 - 116.

5. IDrive. [Електроний ресурс] / Режим доступу: https://www.idrive.com/ (дата звернення: 20.12.2017)

6. Best Cloud Backup Services 2018. [Електроний ресурс] / Режим доступу: https://www.tomsguide.com/us/best-cloud-backup,review-2678.html (дата звернення: 20.12.2017)

7. OneDrive Review. [Електроний ресурс] / Режим доступу: https://www.cloudwards.net/review/onedrive/ (дата звернення: 20.12.2017)

8. Number of registered Dropbox users from April 2011 to March 2016 (in millions). [Електроний ресурс] / Режим доступу: https://www.statista.com/statistics/261820/number-of-registered-dropbox-users/ (дата звернення: 20.12.2017)

9. Google Drive. [Електроний ресурс] / Режим доступу: https://en.wikipedia.org/wiki/Google_Drive (дата звернення: 20.12.2017)

10. Google Drive and IDrive – A Cloud Collaboration Cocktail for 2017 [Електроний ресурс] / Режим доступу: https://www.cloudwards.net/google-drive-vs-idrive/ (дата звернення: 20.12.2017)

11. Apple has over 11 million Apple Music subscribers, 782 million iCloud users. [Електроний ресурс] / Режим доступу: http://www.zdnet.com/article/apple-has-over-11-million-apple-music-subscribers-782-million-icloud-users/ (дата звернення: 20.12.2017)

12. Key-Value Data Store in iCloud [Електроний ресурс] / Режим доступу: https://dzone.com/articles/key-value-data-store-icloud (дата звернення: 20.12.2017)

13. SugarSync Review [Електроний ресурс] / Режим доступу: https://www.lifewire.com/sugarsync-review-2617942 (дата звернення: 20.12.2017)

14. SugarSync Review [Електроний ресурс] / Режим доступу: https://www.cloudwards.net/review/sugarsync/ (дата звернення: 20.12.2017)

15. BackBlaze for Business Review. [Електроний ресурс] / Режим доступу: https://www.cloudwards.net/review/backblaze-for-business/ (дата звернення: 20.12.2017)

16. Carbonite for Office Review. [Електроний ресурс] / Режим доступу: https://www.cloudwards.net/review/carbonite-for-office/ (дата звернення: 20.12.2017)

17. DAS vs NAS vs SAN: Which is best for virtual storage? [Електроний ресурс] / Режим доступу: http://www.computerweekly.com/tip/DAS-vs-NAS-vs-SAN-Which-is-best-for-virtual-storage (дата звернення: 20.12.2017)

18. NAS. [Електроний ресурс] / Режим доступу: http://searchstorage.techtarget.com/definition/network-attached-storage (дата звернення: 20.12.2017)

19. DAS. [Електроний ресурс] / Режим доступу: http://searchstorage.techtarget.com/definition/direct-attached-storage (дата звернення: 20.12.2017)

20. Which is faster SAN or DAS. [Електроний ресурс] / Режим доступу: http://www.sqlteam.com/article/which-is-faster-san-or-directly-attached-storage(дата звернення: 20.12.2017)

21. Eno Thereska. IOFlow: A Software-Defined Storage Architecture. / Eno Thereska, Antony Rowstron, Hitesh Ballani, Tom Talpey, Greg O'Shea, Richard Black, Thomas

Karagiannis, Timothy Zhu // SOSP '13 Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles .– 2013. - PP.182-196 .

22. J. Mace. Retro: Targeted resource management in multi-tenant distributed systems. / J. Mace , P. Bodik, R. Fonseca, M. Musuvathi // USENIX NSDI'15. - 2015. - PP. 589-603.

23. Raúl Gracia-Tinedo. IOStack: Software-Defined Object Storage. / Raúl Gracia-Tinedo, Pedro García-López, Marc Sánchez-Artigas // IEEE Internet Computing. - 2016. - PP. 10-18.

24. Telenyk Sergii. Optimization of data access in tiered storage. / Telenyk Sergii, Bukasov Maxim // 2016 International Conference Radio Electronics & Info Communications (UkrMiCo). – 2016. -PP. 1-4.

25. Теленик С.Ф. Моделі оптимізації багаторівневого зберігання даних. / С. Ф. Теленик, М. М. Букасов, О. К. Карнаухов, В. Ф. Філімонов, М. В. Моргун // Вісник Національного технічного університету України "КПІ". Інформатика, управління та обчислювальна техніка. - 2015. - Вип.3 — С. 48-53.

26. Frank Bunn. Storage Virtualization. / Frank Bunn, Nik Simpson, Robert Peglar, Gene Nagle // Storage Networking Industry Association (SNIA). – 2004. - P. 16.

27. Douglas Santry. Violet: A Storage Stack for IOPS/Capacity Bifurcated Storage Environments. / Douglas Santry, Kaladhar Voruganti // Proceedings of USENIX ATC '14: 2014 USENIX Annual Technical Conference. - 2014. - PP. 13-24.

28. Junhee Ryu. File-System-Level Storage Tiering for Faster Application Launches on Logical Hybrid Disks. / Junhee Ryu, Doungeun Lee, Changhee Han, Heonshik Shin, Kyungtae Kang // IEEE Access (Volume: 4). - 2016. - PP.3688 – 3696.

29. Xiaojian Wu. Exploiting concurrency to improve latency and throughput in a hybrid storage system. / Xiaojian Wu, Narasimha Reddy // Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on. - 2010.

30. Dbench benchmark. [Електроний ресурс] / Режим доступу: ftp://samba.org/pub/tridge/dbench/ (дата звернення: 20.12.2017)

31. NetApp. [Електроний ресурс] / Режим доступу: https://www.netapp.com/us/index.aspx (дата звернення: 20.12.2017)

32. Shun Kaneko, Takaki Nakamura. A guideline for data placement in heterogeneous distributed storage systems. / Shun Kaneko, Takaki Nakamura, Hitoshi Kamei, Hiroaki Muraoka // 2016 5th IIAI International Congress on Advanced Applied Informatics. - 2016. - PP. 942-945.

33. Gong Zhang. Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment. / Gong Zhang, Lawrence Chiu, Ling Liu // Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. - 2010.

34. Giovanni Morana. Self-Managing Distributed Systems and Globally Interoperable Network of Clouds. / Giovanni Morana // The IS4SI 2017 Summit DIGITALISATION FOR A SUSTAINABLE SOCIETY. - Sweden, 2017.

35. Apache HTTP Server. [Електроний ресурс] / Режим доступу: https://httpd.apache.org/(дата звернення: 20.12.2017)

36. П.А. Рахман. Модели надежности отказоустойчивых систем хранения данных. / Рахман П.А., Каяшев А.И., Шарипов М.И. // Вестник УГАТУ. - 2015. - pp. 155-166.

37. Sudharshan S. Vazhkudai. CATCH: A Cloud-based Adaptive Data Transfer Service for HPC / Sudharshan S., Henry M. Monti, Ali R. Butt // Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. - 2011.

38. Microsoft Azure. [Електроний ресурс] / Режим доступу: https://azure.microsoft.com/ (дата звернення: 20.12.2017)

39. Боданюк М.Є.. Управління системами збереження даних. / М.Є. Боданюк , О.К. Карнаухов, О.І. Ролік // Electronics and communications. - 2013. - № 5. - С. 81-90.

40. Z. Yang. AutoTiering: Automatic Data Placement Manager in Multi Tier All-Flash Datacenter. / Z. Yang, M. Hoseinzadeh, A. Andrews, C. Mayers, D. T. Evans, R. T. Bolt, J. Bhimani, N. Mi, and S. Swanson // 36th IEEE International Performance Computing and Communications Conference. — 2017.

41. VMware ESXi. [Електроний ресурс] / Режим доступу: www.vmware.com/products/vsphere-hypervisor.html (дата звернення: 20.12.2017)

42. Z Yang. Automatic and Scalable Data Replication Manager in Distributed Computation and Storage Infrastructure of Cyber-Physical Systems / Z Yang, J Bhimani, J

Wang, D Evans, N Mi // Scalable Computing: Practice and Experience - 2017. - 18 (4) - PP. 291–311

43. Easy Tier Function. [Електроний ресурс] / Режим доступу: https://www.ibm.com/support/knowledgecenter/STVLF4_7.8.0/spectrum.virtualize.780.doc/svc_easy_tier.html (дата звернення: 20.12.2017)

44. MDisks. [Електроний ресурс] / Режим доступу: https://www.ibm.com/support/knowledgecenter/en/STHGUJ_7.3.0/com.ibm.storwize.v5000.730.doc/svc_mdisksovr_1bchav.html (дата звернення: 20.12.2017)

45. EMC VNX – FAST VP explained. [Електроний ресурс] / Режим доступу: http://www.storagefreak.net/2014/06/emc-vnx-fast-vp-explained (дата звернення: 20.12.2017)

46. Storage Vendor Showdown: EMC vs. Pure Part:1. [Електроний ресурс] / Режим доступу: https://www.reliant-technology.com/storage_blog/storage-showdown-emc-pure-part-1/ (дата звернення: 20.12.2017)

47. All-Flash, Solid-Sate Array Data Storage. [Електроний ресурс] / Режим доступу: https://www.hpe.com/us/en/storage/3par.html (дата звернення: 20.12.2017)

48. IBM's Hybrid Cloud Storage Strategy. [Електроний ресурс] / Режим доступу: https://www.virtualtechgurus.com/ibms-hybrid-cloud-storage-strategy/ (дата звернення: 20.12.2017)

49. Drobo. [Електроний ресурс] / Режим доступу: http://www.drobo.com/how-it-works/ (дата звернення: 20.12.2017)

50. Сравнение SSD и HDD дисков в реальных условиях использования. [Електроний ресурс] / Режим доступу: https://geektimes.ru/post/276052/ (дата звернення: 20.12.2017)

51. Consumer SSDs and hard drive prices are nearing parity. [Електроний ресурс] / Режим доступу: https://www.computerworld.com/article/3010395/solid-state-driveshtml (дата звернення: 20.12.2017)

52. Process Monitor. [Електроний ресурс] / Режим доступу: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon (дата звернення: 10.04.2018)

ДОДАТОК А Графічний матеріал

ПЛАКАТ 1 Блок-схема алгоритму міграції даних на сховищі фізичного серверу

ПЛАКАТ 2 Блок-схема алгоритму реплікації даних між фізичними серверами

ПЛАКАТ 3 Математична модель міграції даних на сховищі фізичного серверу

ПЛАКАТ 4 Математична модель реплікації даних між фізичними серверами

ПЛАКАТ 5 Схема структурна класів програмного забезпечення

ПЛАКАТ 6 Приклад оформлення вхідних даних

ПЛАКАТ 7 Результати інтенсивності використання файлів при роботі з операційною системою Windows